webAPI

Day01

课上练习

1.2 获取标签

练习1. 通过id获取到a标签 (可以自己给a添加id)

练习2. 通过标签名 获取到所有h3标签, 并打印

练习3. 通过querySelectAll() 获取所有的li标签

练习4. 通过querySelectAll() 获取所有小字开头的名字的li标签

模板标签可复制

```
<div class="the_div">
    <p>我是p标签</p>
    <h3>我是h3</h3>
    <h3>我是h3</h3>
    <h3>我是h3</h3>
    <a id="theA" href="#">我是一个a标签</a>
    <ul>
        <li>北京</li>
        <li>天津</li>
        <li>东京</li>
    </ul>
    <ul class="my_li">
        <li>小明</li>
        <li>小红</li>
        <li>小蓝</li>
    </ul>
</div>
```

答案

```
<script>
    // 练习1. 通过id获取到a标签 (可以自己给a添加id)
    console.log(document.getElementById("theA"));

    // 练习2. 通过标签名 获取到所有h3标签, 并打印
    console.log(document.querySelector("h3"));

    // 练习3. 通过querySelectAll() 获取所有的li标签
    console.log(document.querySelectorAll("li"));

    // 练习4. 通过querySelectAll() 获取所有小字开头的名字的li标签
    console.log(document.querySelectorAll(".my_li>li"));
</script>
```

2.2 点击事件 - 自定义属性

```
<ul id="myUL">
    <li>我是1</li>
    <li>我是2</li>
    <li>我是3</li>
    <li>我是4</li>
    <li>我是5</li>
    <li>我是6</li>
    <li>我是7</li>
    <li>我是8</li>
    <li>我是9</li>
    <li>我是10</li>
</ul>
<script>
    // 练习1: 点击第一个li标签, 弹出你好
    // // 1. 获取第一个li标签
    // var oneLi = document.querySelectorAll("#myUL>li:first-of-type")[0];
    // // 2. 绑定点击事件
    // oneLi.onclick = function(){
    //     alert("你好");
    // }


    // 练习2: 给所有li标签, 绑定点击事件都能弹出你好
    // // 1. 获取li标签
    // var liList = document.querySelectorAll("#myUL>li");
    // // 2. 产生索引值
    // for (var i = 0; i < liList.length; i++) {
    //     // 3. 用索引值去list集合中取出每个li标签, 绑定点击事件
    //     liList[i].onclick = function(){
    //         // 4. 事件触发后, 要执行的代码
    //         alert("你好");
    //     }
    // }

    // 练习3: 点击某个li, 弹出对应的索引值
    // 1. 获取li标签
```

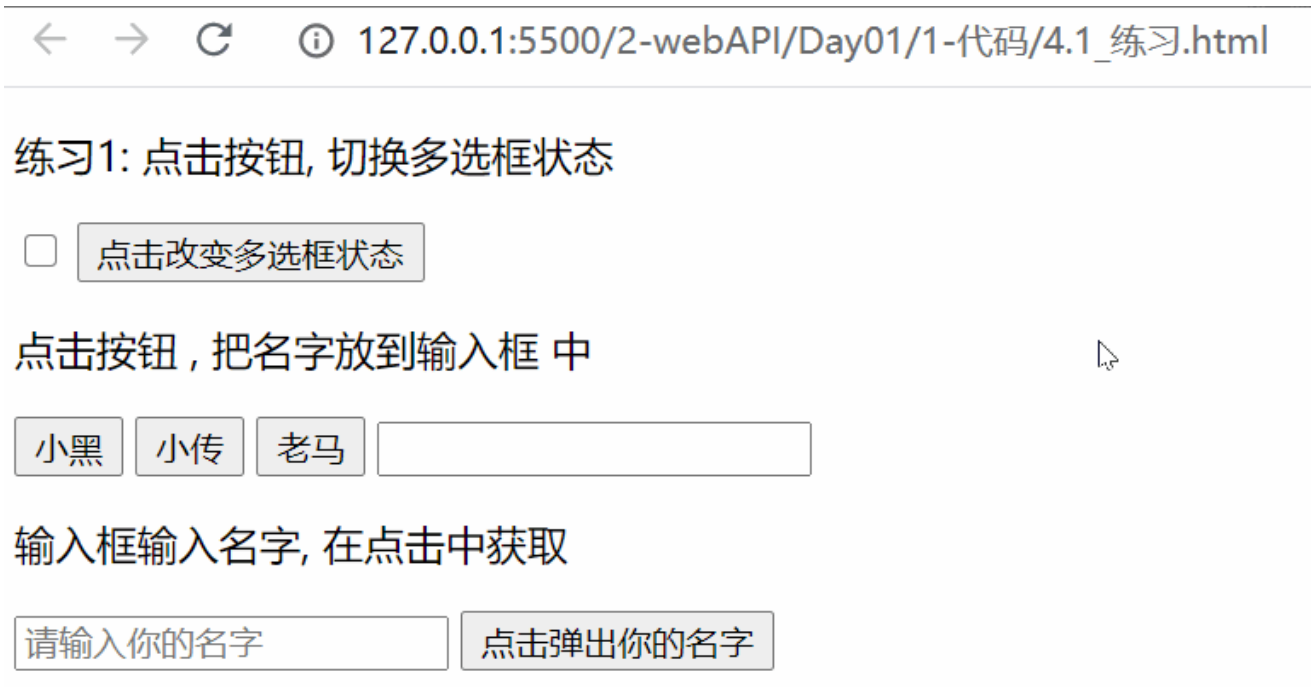
```
    var liList = document.querySelectorAll("#myUL>li");
    // 2. 产生索引值
    for (var i = 0; i < liList.length; i++) {
        // 3. 给每个li扩展属性，保存索引值
        liList[i].index = i;
        // 4. 绑定点击事件
        liList[i].onclick = function(){
            // 5. 访问点击的事件源的index属性的值
            alert(this.index);
        }
    }

    // 问题：为什么点击哪个li都是10呢?
    // 因为:
    // 1. 网页打开，for循环已执行完毕，i的值已为10
    // 2. 点击li才会触发事件函数执行，访问i的值，取出的是10
    // 解决:
    // 3. 让每个li扩展自定义属性，保存自己的索引值
    // 4. 使用this来代表对应的 触发事件的 li标签
```

3.1 事件和属性


```
<style>
    img{
        width: 200px;
    }
    .red_div{
        width: 100px;
        height: 100px;
        background-color: red;
```

```
    }
    .blue_div{
        width: 100px;
        height: 100px;
        background-color: blue;
    }
</style>
</head>
<body>
    <button id="btnOne">练习1的按钮</button>
    <div id="divOne"></div>

    <p id="myP">练习2的p标签</p>

    <button id="btnThree">练习3，点击修改图片</button>
    <img src="./images/a.jpg" alt="" id="imgThree">

    <button id="btnFour">练习4：点击修改下面div背景色</button>
    <div class="red_div" id="divFour"></div>
    <script>
        // 练习1. 点击按钮，修改div的内容
        var theBtn = document.getElementById("btnOne");
        var theDiv = document.getElementById("divOne");
        theBtn.onclick = function(){
            theDiv.innerHTML = "给予内容";
        }


        // 练习2. 点击p标签，修改p标签里的内容
        var theP = document.getElementById("myP");
        theP.onclick = function(){
            theP.innerHTML = "p标签内容被修改";
        }


        // 练习3. 点击按钮，修改img标签上的图片
        var btnThree = document.getElementById("btnThree");
        var imgThree = document.getElementById("imgThree");
        btnThree.onclick = function(){
            imgThree.src = "./images/mm.gif";
        }

        // 练习4. 点击按钮，修改body的背景色
        var btnFour = document.getElementById("btnFour");
        var theDivFour = document.getElementById("divFour");
        btnFour.onclick = function(){
            theDivFour.className = "blue_div";
        }
    </script>
</body>
```

4.1 事件和表单




```
<body>
    <div>
        <p>练习1: 点击按钮, 切换多选框状态</p>
        <input type="checkbox" id="box">
        <button id="btn">点击改变多选框状态</button>
    </div>

    <div>
        <p>点击按钮 , 把名字放到输入框 中</p>
        <input type="button" value="小黑" class="name_btn">
        <input type="button" value="小传" class="name_btn">
        <input type="button" value="老马" class="name_btn">

        <input type="text" id="userInput">
    </div>

    <div>
        <p>输入框输入名字, 在点击中获取</p>
        <input type="text" placeholder="请输入你的名字" id="user">
        <button id="showBtn">点击弹出你的名字</button>
    </div>

    <script>
        // 练习1: 点击按钮, 切换多选框状态
        // 1. 获取元素
        var box = document.getElementById("box");
        var btn = document.getElementById("btn");
        // 2. 绑定点击事件
        btn.onclick = function(){
            // 3. 多选框选中状态

            // if (box.checked === false) {
```

```
            //     box.checked = true;
            // } else {
            //     box.checked = false;
            // }

            box.checked = !box.checked; // 每次点击 取出当前相反状态赋予回去
        }


        // 练习2：有3个按钮，上面分别有名字，点击以后，把名字赋予到一个输入框中显示
        // 1. 获取 元素
        var btnList = document.querySelectorAll(".name_btn");
        var userInput = document.getElementById("userInput");

        // 2. 产生索引绑定点击事件
        for (var i = 0; i < btnList.length; i++) {
            btnList[i].onclick = function(){
                // 3. 从按钮的value属性上取出名字，赋予到目标输入框的value属性上显示在页面
                userInput.value = this.value;
            }
        }

        // 练习3：在输入框输入你的名字，然后点击按钮，弹出你输入的名字
        var user = document.getElementById("user");
        var showBtn = document.getElementById("showBtn");

        showBtn.onclick = function(){
            alert(user.value);
            // 注意：运行网页后 -> 用户输入完value -> 触发事件再获取
        }

    </script>
</body>
```

案例

3.2 案例 – 美女图片选择

素材-在images里看序号

需求:

/Day01/1-代码/3.2_案例_美女切换.html

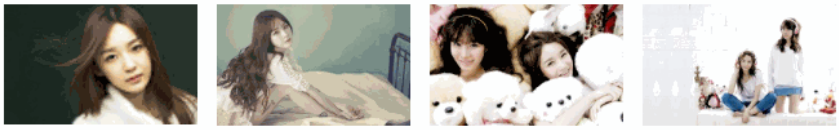

```
<script>
    // 需求：点击按钮 -> 切换美女图片
    // 思路：用索引值对应按钮和标签

    // 1. 获取5个按钮
    var buttonList = document.querySelectorAll(".first>input[type=button]");
    // 2. 获取图片标签
    var theImg = document.getElementById("img");
    // 3. 产生下角标(索引)
    for (var i = 0; i < buttonList.length; i++) {
        // 3.1 每个按钮 - 扩展自定义属性保存索引
        buttonList[i].ind = i;
        // 3.2 每个按钮 - 绑定点击事件
        buttonList[i].onclick = function () {
            // 3.3 拿到索引值 this.ind (this指向onclick前面的人)
            // 核心：用索引值 拼接出 对应的图片地址
            // 3.4 把拼接好的图片地址，赋予到图片标签上
            theImg.src = "./images/3.2/0" + this.ind + ".jpg";
        }
    }
</script>
```

3.3 案例 – 美女画廊

素材在images/看序号

127.0.0.1:5500/2-webAPI/Day01/1-代码/3.3_案例_美女画廊.html

美女画廊



选择一个图片

```
<script>
    // 需求：点击小图，在占位图处显示对应的大图，下面显示美女文字(注：大图和小图是2个图片文件)
    // 思路：索引对应小图和大图，title属性获取美女名字

    // 1. 获取元素
    var liList = document.querySelectorAll(".image_wrap>li"); // 小图的li集合
    var bigImg = document.getElementById("photo"); // 大图img标签
    var nameP = document.getElementById("des"); // 名字容器p标签
    // 2. 产生索引
    for (var i = 0; i < liList.length; i++) {
        // 2.1 每个li - 保存索引
        liList[i].index = i;
        // 2.2 每个li - 绑定点击事件
        liList[i].onclick = function(){
            // 2.3 获取索引: this.index，因为大图从1开始，索引+1
            // 2.4 拼接大图地址，放到大图展示的容器内
            bigImg.src = "./images/3.3/" + (this.index + 1) + ".jpg";
            // 2.5 把美女的名字取出，赋予到p标签上
            nameP.innerHTML = this.title;
        }
    }
</script>
```

4.2 案例 – 全选

需求目标

/Day01/1-代码/4.2_案例_全选.html

标签和样式如下

```
<!DOCTYPE html>
<html>
<head lang="en">
  <meta charset="UTF-8">
  <title>全选 功能</title>
  <style>
    * {
      margin: 0;
      padding: 0;
    }
    table {
      border-collapse: collapse;
      border-spacing: 0;
      border: 1px solid #c0c0c0;
      width: 500px;
      margin: 100px auto;
      text-align: center;
    }
    th {
      background-color: #09c;
      font: bold 16px "微软雅黑";
      color: #fff;
      height: 24px;
    }
    td {
      border: 1px solid #d0d0d0;

      color: #404060;
```

```
      padding: 10px;
    }
  </style>
</head>

<body>
  <table>
    <tr>
      <th>
        <input type="checkbox" name="" id="checkAll" />全选/全不选
      </th>
      <th>菜名</th>
      <th>商家</th>
      <th>价格</th>
    </tr>
    <tr>
      <td>
        <input type="checkbox" name="check" class="ck" />
      </td>
      <td>红烧肉</td>
      <td>隆江猪脚饭</td>
      <td>￥200</td>
    </tr>
    <tr>
      <td>
        <input type="checkbox" name="check" class="ck" />
      </td>
      <td>香酥排骨</td>
      <td>隆江猪脚饭</td>
      <td>￥998</td>
    </tr>
    <tr>
      <td>
        <input type="checkbox" name="check" class="ck" />
      </td>
      <td>北京烤鸭</td>
      <td>隆江猪脚饭</td>
      <td>￥88</td>
    </tr>
  </table>
</body>
</html>
```

正确答案:

```
<script>
    // 需求：点击全选按钮，让下面所有小多选框跟随勾选/未勾选状态
    // 思路：获取全选按钮checked值，赋予给每个小多选框checked属性上

    // 1. 获取标签
    var checkAll = document.getElementById("checkAll");

    var ckList = document.querySelectorAll(".ck");
```

```
    // 2. 给全选多选框绑定点击事件
    checkAll.onclick = function(){
        // 3. 拿到全选框的checked状态 - this.checked
        // 4. 产生索引，取出每个小多选框设置checked
        for (var i = 0; i < ckList.length; i++) {
            ckList[i].checked = this.checked;
        }
    }
</script>
```

4.3 案例 – 小多选框功能

因为这里还没有讲数组的every方法(如果讲了every, 可以用every来统计, 就不用变量接收统计了)

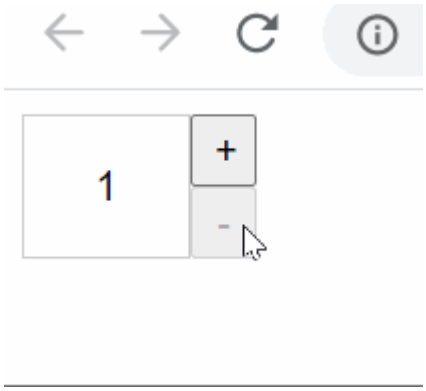
/Day01/1-代码/4.3_案例_小多选框_影响全选.html

全选/全不选	菜名	商家	价格
	红烧肉	隆江猪脚饭	￥200
	香酥排骨	隆江猪脚饭	￥998
	北京烤鸭	隆江猪脚饭	￥88

```
// 需求：点击小多选框，都勾选时，全选框也勾选
// 思路：声明个变量，遍历每个小多选框，如有一个未选中，则变量直接保存false，赋予给全选框，否则全选框设置为true


// 1. 获取标签
var checkAll = document.getElementById("checkAll");
var ckList = document.querySelectorAll(".ck");
checkAll.onclick = function(){
    for (var i = 0; i < ckList.length; i++) {
        ckList[i].checked = this.checked;
    }
}


// 2. 产生索引值
for (var j = 0; j < ckList.length; j++) {
    // 2.1 每个小多选框 - 绑定点击事件
    ckList[j].onclick = function(){

        // 2.2 声明一个变量 - 用于记录小多选框是否有false情况的
```

```
        var checkAllIs;
        // 2.3 (核心) - 无论点击哪个小多选框，都要查看所有小多选框的选中状态
        for (var k = 0; k < ckList.length; k++) {
            // 2.4 如当前小多选框未选中，checked值为false，那么checkAllIs就应该 是false
            if (ckList[k].checked === false) {
                checkAllIs = false;
            }
        }

        // 3. for循环执行完毕，checkAllIs为false，证明有小多选框是未选中的，否则就是都选中，把最后的结果赋予给全选的多选框checked
        checkAll.checked = (checkAllIs === false ? false : true);
    }
}

// 更好的做法
// 1. 获取标签
var checkAll = document.getElementById("checkAll");
var ckList = document.querySelectorAll(".ck"); // 所有小多选框

// 2. 全选影响所有小的
checkAll.onclick = function(){
    var allChecked = this.checked;
    Array.from(ckList).map(function(el){
        el.checked = allChecked;
    })
}

// 3. 小影响多
var ckArr = Array.from(ckList);
ckArr.map(function(el){
    el.onclick = function(){
        // 查找数组里不符合条件的，直接返回false(停止循环)
        var isAll = ckArr.every(function(el){return el.checked == true}); // 筛选是否有不符合条件的返回false
        checkAll.checked = isAll == false ? false : true;
    }
})
```

4.4 案例 – 商品数量控制



标签和样式复制

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>商品数量控制</title>
  <style>
    div {
      width: 80px;
    }

    input[type=text] {
      width: 50px;
      height: 44px;
      outline: none;
      border: 1px solid #ccc;
      text-align: center;
    }

    input[type=button] {
      height: 24px;
      width: 22px;
    }

    input {
      float: left;
    }
  </style>
</head>

<body>
  <div>
    <input type="text" id="total" value="1" readonly>
    <input type="button" value="+" id="add">
    <input type="button" value="-" id="reduce" disabled>
  </div>
</body>
```

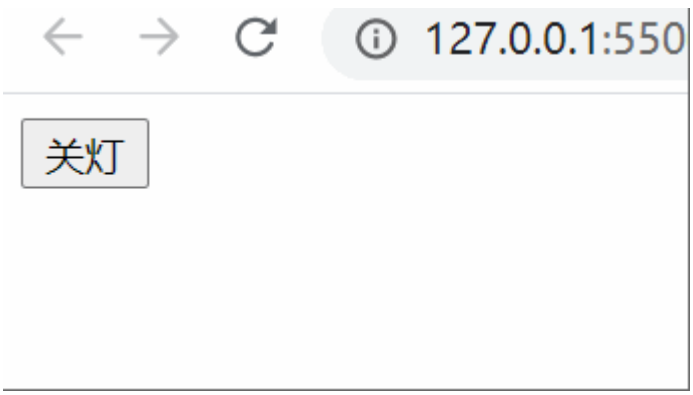
```
</html>
```

正确代码:js

```
// 需求：点击+，增加数量，点击-，减少数量，当数量为1时，禁用-号
// 思路：实现+和-按钮功能，分别判断-按钮的禁用状态

// 1. 获取标签
var total = document.getElementById("total");
var addBtn = document.getElementById("add");
var reduceBtn = document.getElementById("reduce");
// 2. 实现+的功能
// 注意：表单标签获取的值，都是字符串类型
addBtn.onclick = function () {
   // 2.1 转换
   var theValue = parseInt(total.value);
   // 2.2 +1 赋予到标签上
   total.value = theValue + 1;
   // 2.3 取消-法的禁用状态
   reduceBtn.disabled = false;
}

// 3. 实现-法的功能
reduceBtn.onclick = function () {
   // 3.1 实现-1后需要回标签上
   total.value = total.value - 1;
   // 3.2 如果值等于1，就开启-按钮的禁用状态
   if (total.value == 1) {
       reduceBtn.disabled = true;
  }
}
```

作业

作业1 - 点击开关灯



需求1: 状态判断, innerHTM页面上已知有一个按钮, 按钮文字显示"关灯", 单击切换成"开灯"字, 在"开灯"和"关灯" 字之间点击来回切换的效果即可. 请实现效果

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Day01_作业讲解</title>
    </head>
    <body>
        <button id="myBtn">关灯</button>
        <script>
            // 需求：按钮默认显示"关灯"文字，点击按钮，切换文字为"开灯"，再点击切换为"关灯"，实现切换
效果
            // 思路：绑定点击事件，通过判断当前内容，来设置应该显示的内容

            // 方式1：通过标签内容来判断
            // 1. 获取元素
            // var myBtn = document.getElementById("myBtn");
            // // 2. 绑定点击事件
            // myBtn.onclick = function(){
            // // 3. (核心)- 如何判断当前内容应该是开灯 / 关灯
            // if (this.innerHTML === "关灯") {
            // this.innerHTML = "开灯";
            // } else {
            // // 开灯 -> 关灯
            // this.innerHTML = "关灯";
            // }

            // // 上边if和else，优化后：
            // // this.innerHTML = (this.innerHTML === "关灯") ? "开灯" : "关灯";
            // }

            // 方式2：使用标记变量来判断
            // var myBtn = document.getElementById("myBtn");
            // // 1. 准备标记变量
            // var flag = false; // (false代表关灯文字)
            // myBtn.onclick = function(){
            // // 2. 用标记变量代替文字判断
            // if (flag === false) {
            // this.innerHTML = "开灯";
            // flag = true; // (true代表当前页面文字为开灯)
            // } else {
            // this.innerHTML = "关灯";
            // flag = false;
            // }

            // // if和else优化
            // // this.innerHTML = (flag === false) ? "开灯" : "关灯";
            // // this.innerHTML = flag ? "关灯" : "开灯";
            // // flag = !flag; // 每次flag都取反
            // }

            // 方式3：(了解) 用奇偶值
            var myBtn = document.getElementById("myBtn");
```

```
            // 1. 计数变量
            var count = 0;
            myBtn.onclick = function(){
                // 2. 每次点击按钮 count都+1
                count++;
                // 3. 根据奇偶值来判断当前是开灯/关灯
                if (count % 2 !== 0) {
                    this.innerHTML = "开灯";
                } else {
                    this.innerHTML = "关灯";
                }
            }
        </script>
    </body>
</html>
```

Day02

课上练习

暂无

案例

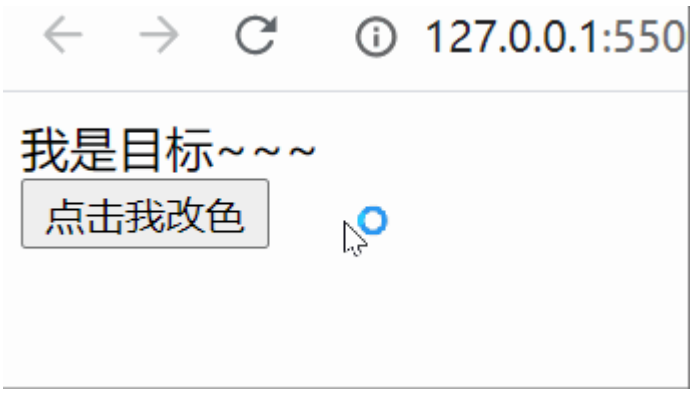
5.2 – 重构美女画廊

效果与昨天相同 - 就是改变js代码 - 昨天用索引拼接图片名路径, 今天用数组下标换图片路径

```
// 方式1: 在"标签上"添加自定义属性(对应的大图地址)（手写）（了解）
// 1. 获取标签
// var liList = document.querySelectorAll(".image_wrap>li");
// var photo = document.getElementById("photo");
// // 2. 产生索引
// for (var i = 0; i < liList.length; i++) {
//     // 2.1 每个li - 绑定点击事件
//     liList[i].onclick = function(){
//         // 2.2 点击的li - 取出自定义属性保存的大图地址，赋予给大图img
//         photo.src = this.getAttribute("bigS");
//     }
// }

// 方式2: 在"js"添加自定义属性（常用）
// 1. 定义大图数组 - 分配个每个li标签
var bigImgArr = ["./images/5.2/1.jpg", "./images/5.2/2.jpg", "./images/5.2/3.jpg",
"./images/5.2/4.jpg"];
// 2. 获取li标签集合
var liList = document.querySelectorAll(".image_wrap>li");
// 3. 产生索引 - 为li绑定大图数据

for (var i = 0; i < liList.length; i++) {
```

```
    // 3.1 给每个li - 自定义属性bigSrc = 对应大图地址
    liList[i].bigSrc = bigImgArr[i];
}
// 4. 给li绑定点击事件
for (var j = 0; j < liList.length; j++) {
    liList[j].onclick = function(){
        // 4.1. 取出自定义属性上大图地址，赋予到目标上
        photo.src = this.bigSrc;
    }
}
```

6.3 案例 - 点击按钮 - 随机变色




```
// 需求：点击按钮，随机改变div背景色
// 思路：点击按钮时，随机颜色可通过随机数+rgb()字符串设置给标签样式上

// 1. 获取标签
var myBtn = document.getElementById("btn");
var myDiv = document.getElementById("myDiv");

// 2. 绑定点击事件
myBtn.onclick = function(){
    // 2.1 产生r，g，b三个变量的颜色随机数字
    var r = Math.floor(Math.random() * (255 - 0 + 1) + 0);
    var g = Math.floor(Math.random() * (255 - 0 + 1) + 0);
    var b = Math.floor(Math.random() * (255 - 0 + 1) + 0);
    // 2.2 产生rgb颜色字符串
    var rgbStr = "rgb(" + r + ", " + g + ", " + b + ")";
    // 2.3 改变div颜色样子
    myDiv.style.backgroundColor = rgbStr;
}
```

6.4 案例 - 开关灯(双按钮)

查看标签所有样式的方式: 检查 -> Elements -> Computed -> (勾选show all)

```
// 需求：点击开灯网页亮，点击关灯，网页黑
// 思路：点击关灯，改变背景为黑色，点击开灯，恢复原来颜色

// 1. 获取标签，绑定点击事件
var openBtn = document.getElementById("openBtn");
var closeBtn = document.getElementById("closeBtn");

// 2. 关灯按钮事件
closeBtn.onclick = function(){
    // 2.1 改变body背景色
    document.body.style.backgroundColor = "black";
}

// 3. 开灯按钮事件
openBtn.onclick = function(){
    // 3.1 改变body背景色
    // 查看body初始值发现背景色为rgba(0, 0, 0, 0); 白色全透明
    // 设置时，可使用空字符串，transparent字符串，rgba(0, 0, 0, 0)
    document.body.style.backgroundColor = "";
}
```

6.5 案例 - 开关灯(单按钮)


```
// 需求：点击按钮，自动判断当前状态，取相反状态显示
// 思路：准备flag保存文字状态，点击时，通过flag判断，要设置的样式和内容，同时更新flag值

// 1. 获取标签
var btn = document.getElementById("btn");
// 2. flag标记变量 - 保存文字状态(false -> 现在是关灯文字)
var flag = false;

// 3. 按钮点击事件
```

```
btn.onclick = function(){
   // 3.1 当前文字为关灯
   if (flag === false) {
       // 3.2 设置背景色黑色
       document.body.style.backgroundColor = "black";
       // 3.3 修改按钮
       this.innerHTML = "开灯";
       // 3.4 更新flag状态true（代表文字开灯）
       flag = true;
  } else {
       // 3.5 用户想开灯，设置背景色透明
       document.body.style.backgroundColor = "";
       // 3.6 设置按钮内容
       this.innerHTML = "关灯";
       // 3.7 更新flag状态false（代表文字关灯）
       flag = false;
  }
}

// 优化后
btn.onclick = function(){
   document.body.style.backgroundColor = flag ? "" : "black";
   btn.innerHTML = flag ? "关灯" : "开灯";
   flag = !flag;
}
```

6.6 案例 - 点谁谁亮


```
// 需求：点击某个按钮，按钮高亮，其余人都不亮
// 思路：点击时，排他思想，把所有人都设置成不亮的，再单独对点击的设置亮的

// 1. 获取标签
var btnList = document.querySelectorAll("#myWrap>input[type=button]");
// 2. 产生索引
for (var i = 0; i < btnList.length; i++) {
    // 2.1 每个按钮 - 点击事件
    btnList[i].onclick = function(){
        // 2.2 产生索引，把所有的按钮背景色都干掉
        for (var j = 0; j < btnList.length; j++) {
            btnList[j].style.backgroundColor = "";
        }
        // 2.3 再对单独点击的按钮，设置背景色

        this.style.backgroundColor = "red";
```

```
    }
}
```

6.7 案例 - 点击tab栏切换

图片素材在images/下看序号

模板代码:

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>案例_tab导航切换</title>
    <style>
        * {
            margin: 0;
            padding: 0;
        }
        ul {
            list-style: none;
        }
        .container {
            width: 1120px;
            margin: 0 auto;
        }
        .tab {
            width: 1004px;
            margin: 0 auto;
            border: 1px solid #aaa;
        }
        .tab-control {
            overflow: hidden;
        }

        .tab-control li {
```

```
            width: 200px;
            float: left;
            background-color: #ccc;
            border-top: 3px solid #aaa;
            font: bold 22px/40px "宋体";
            text-align: center;
            cursor: pointer;
            color: #000;
        }
        .tab-control li.active {
            border-top: 3px solid red;
            background-color: #fff;
            color: red;
        }
        .tab-control li+li {
            border-left: 1px solid #aaa;
        }
        .tab-content {
            width: 1000px;
        }
        .tab-content .item {
            display: none;
        }
        .tab-content .item:first-child {
            display: block;
        }
        .tab-content .item img {
            width: 100%;
        }
    </style>
</head>
<body>
    <!--版心-->
    <div class="container">
        <!--tab切换模块-->
        <div class="tab">
            <!--切换按钮-->
            <ul class="tab-control">
                <li class="active">疯狂抢购</li>
                <li>热卖推荐</li>
                <li>猜你喜欢</li>
                <li>新品上架</li>
                <li>整箱特卖</li>
            </ul>
            <!--切换项-->
            <div class="tab-content" id="tab_content">
                <div class="item">
                    <img src="./images/6.7/01.bmp" />
                </div>
                <div class="item">
                    <img src="./images/6.7/02.bmp" />
                </div>

                <div class="item">
```

```
                    <img src="./images/6.7/03.bmp" />
                </div>
                <div class="item">
                    <img src="./images/6.7/04.bmp" />
                </div>
                <div class="item">
                    <img src="./images/6.7/05.bmp" />
                </div>
            </div>
        </div>
    </div>
</body>
</html>
```

正确js代码

```
// 需求：点击顶部导航，点击高亮，然后同时切换底部div内容
// 思路：顶部导航点谁，给谁设置active的类名，先让所有div先隐藏，再让导航点击索引，找到对应的div让其出现

// 1. 获取标签
var liList = document.querySelectorAll(".tab-control>li");
var itemList = document.querySelectorAll("#tab_content>.item");
// 2. 产生索引
for (var i = 0; i < liList.length; i++) {
    // 2.1 每个li导航 - 绑定索引值
    liList[i].index = i;
    // 2.2 每个li导航 - 点击事件
    liList[i].onclick = function(){
        // 2.3 移出所有li导航的class，和所有div都先隐藏
        for (var j = 0; j < liList.length; j++) {
            liList[j].className = "";
            itemList[j].style.display = "none";
        }
        // 2.4 你点击的li高亮，和索引对应的div出现
        this.className = "active";
        itemList[this.index].style.display = "block";
    }
}
```

7.1 案例 - 搜索提示列表

图片素材在images/看序号

00/2-webAPI/Day02/1-代码/7.1_案例_搜索提示列表.html

标签模板

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>搜索提示列表</title>
  <style>
    * {
      margin: 0;
      padding: 0;
    }
    .box {
      position: absolute;
      top: 100px;
      left: 300px;
    }
    input {

      box-sizing: border-box;
```

```
      width: 300px;
      height: 56px;
      background-image: url('./images/7.1/search.png');
      padding: 4px 6px;
      border: 0 none;
      outline: 0 none;
      font-size: 24px;
    }
    .list {
      border: 1px solid #ccc;
      position: absolute;
      top: 54px;
      left: 2px;
      display: none;
    }
    .show {
      display: block;
    }
  </style>
</head>
<body>
  <div class="box">
    <input type="text" id="search">
    <div class="list" id="list">
      <img src="./images/7.1/mi.png" alt="">
    </div>
  </div>
  <script>

  </script>
</body>
</html>
```

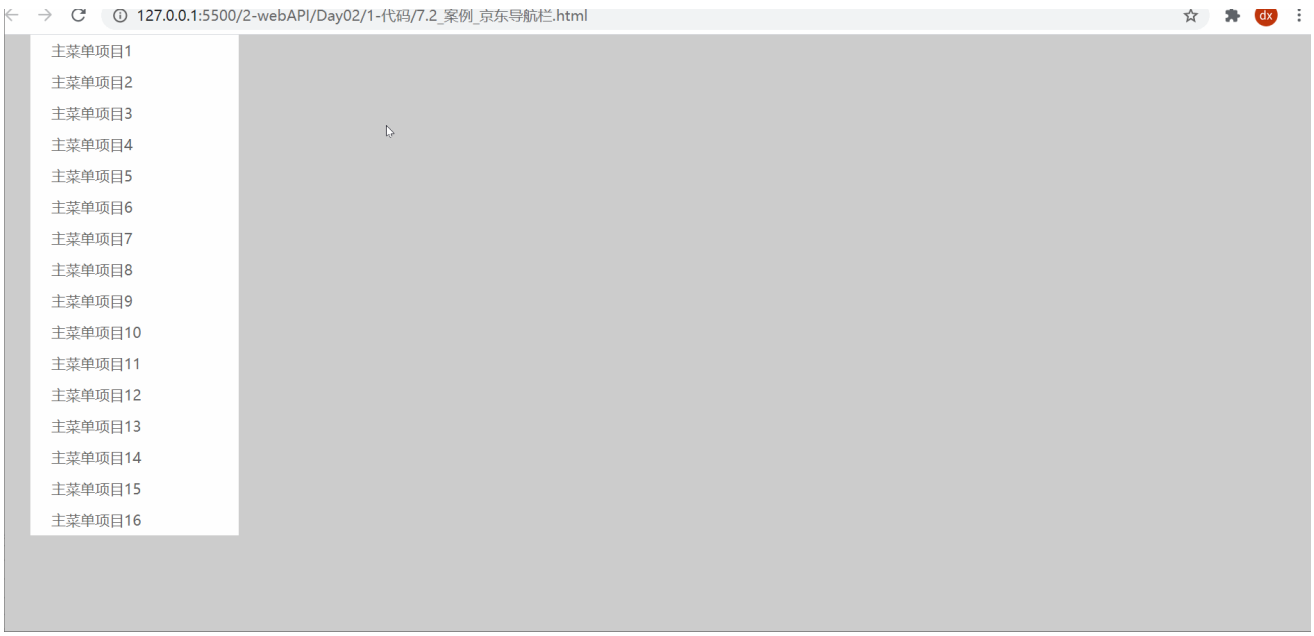
正确js代码

```
// 需求：点击输入框，显示联想列表，失去焦点时，联想列表消失
// 思路：给输入框绑定onfocus和onblur事件，在相应位置中，设置联想列表显示/隐藏

// 1. 获取标签
var theSearch = document.getElementById("search");
var imgDiv = document.getElementById("list");

// 2. 给输入框绑定onfocus事件
theSearch.onfocus = function(){
    imgDiv.style.display = "block"; // 让下拉列表显示
}
// 3. 输入框绑定onblur事件
theSearch.onblur = function(){
    imgDiv.style.display = "none"; // 让下拉列表隐藏
}
```

7.2 案例 - 京东导航栏

素材在images下看序号

效果如下:



标签模板

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <style>
    * {
      margin: 0;
      padding: 0;
    }
    body {
      background: #ccc;
    }
    ul {
      list-style: none;
    }
    .w {
      width: 1200px;
      margin: 0 auto;
    }
    a {
      color: #666;
      text-decoration: none;
    }
    a:hover {
      color: red;
    }
    .slide {
      position: relative;
      background-color: red;
    }
```

```
    /* 左侧 */
    .left {
      width: 200px;
      background: #fff;
      color: #ccc;
      position: absolute;
    }
    /* 左侧的li */
    .left ul li {
      padding-left: 20px;
      line-height: 30px;
      font-size: 14px;
      cursor: pointer;
    }
    /* 左侧li被选中的样式 */
    .left ul li.active {
      background: rgba(153, 153, 153, 0.657);
    }
    /* 右侧详细菜单 */
    .right {
      position: absolute;
      left: 200px;
    }
    .right .item {
      display: none;
      border-left: 1px solid #eee;
    }
    .right .item.active {
      display: block;
    }
  </style>
</head>

<body>
  <!-- 版心 -->
  <div class="w">
    <div class="slide">
      <!-- 左侧 -->
      <div class="left">
        <ul>
          <li><a href="#">主菜单项目1</a></li>
          <li><a href="#">主菜单项目2</a></li>
          <li><a href="#">主菜单项目3</a></li>
          <li><a href="#">主菜单项目4</a></li>
          <li><a href="#">主菜单项目5</a></li>
          <li><a href="#">主菜单项目6</a></li>
          <li><a href="#">主菜单项目7</a></li>
          <li><a href="#">主菜单项目8</a></li>
          <li><a href="#">主菜单项目9</a></li>
          <li><a href="#">主菜单项目10</a></li>
          <li><a href="#">主菜单项目11</a></li>
          <li><a href="#">主菜单项目12</a></li>

          <li><a href="#">主菜单项目13</a></li>
```

```
          <li><a href="#">主菜单项目14</a></li>
          <li><a href="#">主菜单项目15</a></li>
          <li><a href="#">主菜单项目16</a></li>
        </ul>
      </div>
      <!-- 右侧 -->
      <div class="right">
        <div class="item">
          <img src="./images/7.2/01.png"></div>
        <div class="item">
          <img src="./images/7.2/02.png"></div>
        <div class="item">
          <img src="./images/7.2/03.png"></div>
        <div class="item">
          <img src="./images/7.2/04.png"></div>
        <div class="item">
          <img src="./images/7.2/05.png"></div>
        <div class="item">
          <img src="./images/7.2/06.png"></div>
        <div class="item">
          <img src="./images/7.2/07.png"></div>
        <div class="item">
          <img src="./images/7.2/08.png"></div>
        <div class="item">
          <img src="./images/7.2/09.png"></div>
        <div class="item">
          <img src="./images/7.2/10.png"></div>
        <div class="item">
          <img src="./images/7.2/11.png"></div>
        <div class="item">
          <img src="./images/7.2/12.png"></div>
        <div class="item">
          <img src="./images/7.2/13.png"></div>
        <div class="item">
          <img src="./images/7.2/14.png"></div>
        <div class="item">
          <img src="./images/7.2/15.png"></div>
        <div class="item">
          <img src="./images/7.2/16.png"></div>
      </div>
    </div>
  </div>
  <script>
  </script>
</body>

</html>
```

正确js代码:

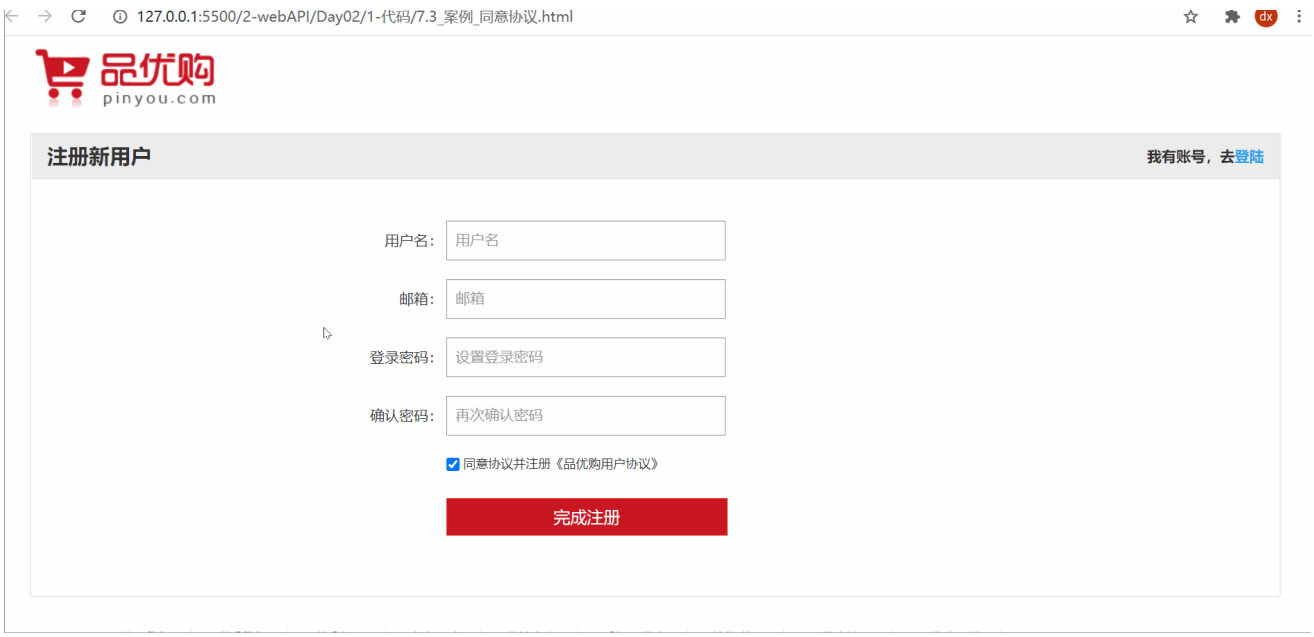
```
// 需求：鼠标移入左侧导航，显示相应右侧导航，而且鼠标也能在右侧分类中滑动，当离开父级标签区域后，导航高
```

```
亮消失,分类隐藏
// 思路：鼠标移入左侧导航，排他思想，只设置当前li高亮和索引对应div右侧出现，移出父级区域时，把导航和div回归初始状态

// 1. 获取相关标签
var liList = document.querySelectorAll(".left>ul>li");
var itemList = document.querySelectorAll(".right>.item");
var slide = document.querySelectorAll(".slide")[0];

// 2. 产生索引
for (var i = 0; i < liList.length; i++) {
    // 2.1 每个li导航 - 绑定索引值
    liList[i].index = i;
    // 2.2 每个li导航 - 移入事件
    liList[i].onmouseenter = function(){
        // 2.3 把所有li和对应div，消除高亮样式和隐藏掉
        for (var j = 0; j < liList.length; j++) {
            liList[j].className = "";
            itemList[j].style.display = "none";
        }
        // 2.4 只对当前移入的li设置高亮样式，和对应索引的div出现
        this.className = "active";
        itemList[this.index].style.display = "block";
    }
}
// 3. 父级slide上移出事件 - (注意，虽然slide没有高度，但是有子级定位元素，从定位元素移出也算从父级区域移出)
slide.onmouseleave = function(){
    // 3.1 把所有li和对应div，消除高亮样式和隐藏掉
    for (var j = 0; j < liList.length; j++) {
        liList[j].className = "";
        itemList[j].style.display = "none";
    }
}
```

7.3 案例 - 同意协议

素材在images/看序号

需求效果如下:



标签模板复制

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
    <title>个人注册</title>
    <link rel="stylesheet" href="./css/7.3/common.css">
    <link rel="stylesheet" href="./css/7.3/index.css">
    <link rel="icon" href="./images/7.3/favicon.ico">
</head>

<body>
    <div class="register py-container ">
        <!--head-->
        <div class="logoArea">
            <a href="home.html" class="logo"></a>
        </div>
        <!--register-->
        <div class="registerArea">
            <h3>注册新用户<span class="go">我有账号，去<a href="login.html">登陆</a></span>
            </h3>
            <div class="info">
                <form class="sui-form form-horizontal">
                    <div class="control-group">
                        <label class="control-label">用户名：</label>
                        <div class="controls">
                            <input type="text" class="input-xfat input-xlarge"
                                placeholder="用户名">
                        </div>
                    </div>
                    <div class="control-group">
                        <label class="control-label">邮箱：</label>

                        <div class="controls">
```

```
                        <input type="text" class="input-xfat input-xlarge"
                            placeholder="邮箱">
                    </div>
                </div>
                <div class="control-group">
                    <label class="control-label">登录密码：</label>
                    <div class="controls">
                        <input type="password"
                            class="input-xfat input-xlarge"
                            placeholder="设置登录密码">
                    </div>
                </div>
                <div class="control-group">
                    <label class="control-label">确认密码：</label>
                    <div class="controls">
                        <input type="password"
                            class="input-xfat input-xlarge"
                            placeholder="再次确认密码">
                    </div>
                </div>
                <div class="control-group">
                    <span class="control-label"> </span>
                    <label class="controls">
                        <input id="agree" name="m1" type="checkbox" checked>
                        <span>同意协议并注册《品优购用户协议》</span>
                    </label>
                </div>
                <div class="control-group">
                    <span class="control-label"></span>
                    <div class="controls btn-reg">
                        <button id="registerBtn"
                            class="sui-btn btn-block btn-xlarge btn-danger"
                            href="#">完成注册</button>
                    </div>
                </div>
            </form>
            <div class="clearfix"></div>
        </div>
    </div>
    <!--foot-->
    <div class="py-container copyright">
        <ul>
            <li><a href="#">关于我们</a></li>
            <li><a href="#">联系我们</a></li>
            <li><a href="#">联系客服</a></li>
            <li><a href="#">商家入驻</a></li>
            <li><a href="#">营销中心</a></li>
            <li><a href="#">手机品优购</a></li>
            <li><a href="#">销售联盟</a></li>
            <li><a href="#">品优购社区</a></li>
            <li><a href="#">品优购公益</a></li>
            <li><a href="#">English Site</a></li>

            <li><a href="#">Contact U</a></li>
```

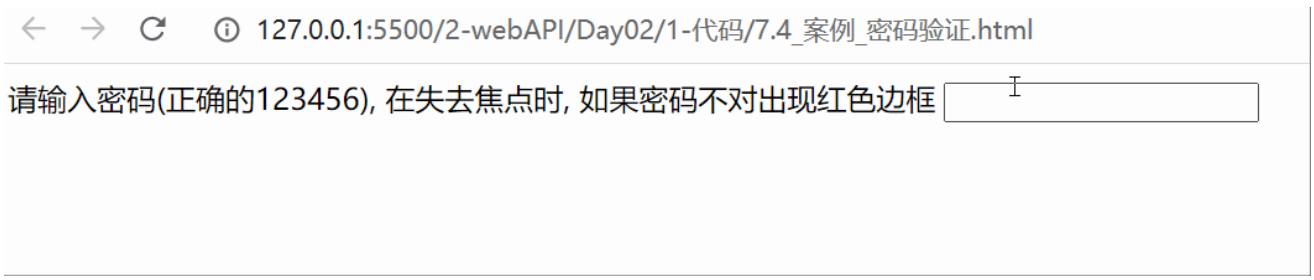
```
            </ul>
            <div class="address">地址：北京市昌平区建材城西路金燕龙办公楼一层 邮编：100096
                电话：400-618-4000 传真：010-82935100</div>
            <div class="beian">京ICP备08001421号京公网安备110108007702</div>
        </div>
    </div>
    <script>
    </script>
</body>


</html>
```

正确js代码:

```
// 需求：点击同意协议前的多选框，勾选时，下面的按钮是可以交互的，未勾选时，是无法交互的
// 思路：给多选框绑定点击事件，获取checked值，来影响下面按钮的disabled值

// 1. 获取标签
var checkedInput = document.getElementById("agree");
var registerBtn = document.getElementById("registerBtn");

// 2. 绑定点击事件
checkedInput.onclick = function(){
    // 2.1 获取当前多选框的状态
    var nowChecked = checkedInput.checked;
    // 2.2 根据多选框来影响按钮disabled效果
    if (nowChecked === true) {
        registerBtn.disabled = false; // 取消禁用
    } else {
        registerBtn.disabled = true; // 开启禁用
    }
}

// 学会如何通过现有的class，找到别人写好的class中的样式 8208和8391
```

7.4 案例 - 密码验证

效果如下:




```
// 需求：当用户输入密码后，失去焦点时，判断，假设正确是123456，输入错误边框红色，否则边框默认即可
// 思路：监听输入框onchange事件，在判断value值是否正确，设置边框相应样式
```

```
// 1. 获取标签
var thePass = document.getElementById("myPass");
// 2. 绑定onchange事件
thePass.onchange = function(){
    // 2.1 判断密码正确 / 没有输入密码
    if (this.value == 123456 || this.value == "") {
        // 2.2 密码正确移除边框
        this.style.border = "";
    } else {
        // 2.3 密码错误，添加红色边框
        this.style.border = "1px solid red";
    }
}
```

作业

作业1 - 购物车 - 数量编辑

images下素材

500/2-webAPI/Day03/1-代码/Day02_作业讲解.html

全选	商品	单价	商品数量	小计	操作
☐	 牛奶	5￥	- 1 +	5￥	删除
☐	 牛奶	10￥	- 2 +	20￥	删除
☐	 牛奶	20￥	- 2 +	40￥	删除
☐	 牛奶	35￥	- 2 +	70￥	删除

删除所选商品　　清理购物车　　已经选中0件商品;总价：0￥　　去结算

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>购物车全选功能</title>

  <!-- 引入初始化 -->
```

```
<style>
  * {
    margin: 0;
    padding: 0;
  }

  ul {
    list-style: none;
  }

  a {
    text-decoration: none;
    color: #666;
  }

  body {
    background: #fff;
    color: #666;
    font-size: 14px;
  }

  input {
    outline: none;
  }

  .clearfix::before,
  .clearfix::after {
    content: '';
    display: block;
    clear: both;
  }

  .clearfix {
    *zoom: 1;
  }
</style>
<!-- 引入购物车样式 -->
<style>
  table {
    width: 800px;
    margin: 0 auto;
    border-collapse: collapse;
  }

  th {
    font: normal 14px/50px '宋体';
    color: #666;
  }

  th,
  td {
    border: none;

    text-align: center;
```

```
    border-bottom: 1px dashed #ccc;
  }

  input[type=checkbox] {
    width: 13px;
    height: 13px;
  }

  tbody p {
    position: relative;
    bottom: 10px;
  }

  tbody .add,
  tbody .reduce {
    float: left;
    width: 20px;
    height: 20px;
    border: 1px solid #ccc;
    text-align: center;

  }

  tbody input[type=text] {
    width: 50px;
    float: left;
    height: 18px;
    text-align: center;
  }

  tbody .count-c {
    width: 98px;
    margin: 0 auto;
  }

  .disabled {
    color: #ddd;
    cursor: not-allowed;
  }

  tbody tr:hover {
    background: rgba(241, 209, 149, 0.945);
  }

  tbody tr.active {
    background: rgba(241, 209, 149, 0.945);
  }

  .controls {
    width: 790px;
    margin: 10px auto;
    border: 1px solid #ccc;

    line-height: 50px;
```

```
      padding-left: 10px;
      position: relative;
    }

    .controls .del-all,
    .controls .clear {
      float: left;
      margin-right: 50px;
    }

    .controls p {
      float: right;
      margin-right: 100px;
    }

    .controls span {
      color: red;
    }

    .controls .pay {
      position: absolute;
      right: 0;
      width: 80px;
      height: 54px;
      background: red;
      font: bold 20px/54px '宋体';
      color: #fff;
      text-align: center;
      bottom: -1px;
    }

    .controls .total-price {
      font-weight: bold;
    }
  </style>
</head>

<body>
  <div class="car">
    <table>
      <thead>
        <tr>
          <th>
            <input type="checkbox" id="all">全选
          </th>
          <th>
            商品
          </th>
          <th>
            单价
          </th>
          <th>

            商品数量
```

```
          </th>
          <th>
            小计
          </th>
          <th>
            操作
          </th>
        </tr>
      </thead>
      <tbody id="carBody">
        <tr>
          <td>
            <input class="s_ck" type="checkbox" readonly>
          </td>
          <td>
            <img src="./images/01.jpg">
            <p>牛奶</p>
          </td>
          <td class="price">
            5￥
          </td>
          <td>
            <div class="count-c clearfix">
              <a href="javascript:" class="reduce disabled">-</a>
              <input type="text" value="1">
              <a href="javascript:" class="add">+</a>
            </div>
          </td>
          <td class="total">
            5￥
          </td>
          <td>
            <a href="javascript:" class="del">删除</a>
          </td>

        </tr>
        <tr>
          <td>
            <input class="s_ck" type="checkbox">
          </td>
          <td>
            <img src="./images/01.jpg">
            <p>牛奶</p>
          </td>
          <td class="price">
            10￥
          </td>
          <td>
            <div class="count-c clearfix">
              <a href="javascript:" class="reduce">-</a>
              <input type="text" readonly value="2">
              <a href="javascript:" class="add">+</a>

            </div>
```

```
      </td>
      <td class="total">
        20￥
      </td>
      <td>
        <a href="javascript:" class="del">删除</a>
      </td>

    </tr>
    <tr>
      <td>
        <input class="s_ck" type="checkbox">
      </td>
      <td>
        <img src="./images/01.jpg">
        <p>牛奶</p>
      </td>
      <td class="price">
        20￥
      </td>
      <td>
        <div class="count-c clearfix">
          <a href="javascript:" class="reduce">-</a>
          <input type="text" readonly value="2">
          <a href="javascript:" class="add">+</a>
        </div>
      </td>
      <td class="total">
        40￥
      </td>
      <td>
        <a href="javascript:" class="del">删除</a>
      </td>

    </tr>
    <tr>
      <td>
        <input class="s_ck" type="checkbox">
      </td>
      <td>
        <img src="./images/01.jpg">
        <p>牛奶</p>
      </td>
      <td class="price">
        35￥
      </td>
      <td>
        <div class="count-c clearfix">
          <a href="javascript:" class="reduce">-</a>
          <input type="text" readonly value="2">
          <a href="javascript:" class="add">+</a>
        </div>

      </td>
```

```
          <td class="total">
            70￥
          </td>
          <td>
            <a href="javascript:" class="del">删除</a>
          </td>

        </tr>

      </tbody>
    </table>
    <div class="controls clearfix">
      <a href="javascript:" class="del-all">删除所选商品</a>
      <a href="javascript:" class="clear">清理购物车</a>
      <a href="javascript:" class="pay">去结算</a>
      <p>
        已经选中<span id="totalCount">0</span>件商品;总价：<span id="totalPrice" class="total-
price">0￥</span>
      </p>
    </div>
  </div>
  <script>


  </script>
</body>

</html>
```

js代码

```
// 需求：实现全选功能，实现增减功能，实现计算单价功能(练习标签关系查找)
    // 思路：全选和小多选影响全选功能，点击+和-按钮控制数量，+和-同时计算小计的价格
    // 1. 获取元素
    var all = document.getElementById("all"); // 全选多选框
    var sckList = document.querySelectorAll(".s_ck"); // 获取所有小多选框
    var reduceList = document.querySelectorAll(".reduce"); // 减法按钮(多个)
    var numInputList = document.querySelectorAll(".count-c>input[type=text]"); // 获取商品数量标
签
    var addList = document.querySelectorAll(".add"); // 加法按钮(多个)[a, a, a, a]
    var priceTdList = document.querySelectorAll(".price"); // 单价Td集合
    var totalTdList = document.querySelectorAll(".total"); // 小计Td集合

    // readonly 意思：不能在页面上手动输入内容
    // 2. 全选 影响-> 小多选功能
    all.onclick = function () {
      // 2.1 产生索引值
      for (var i = 0; i < sckList.length; i++) {
        // 2.2 使用索引值换出每个小多选框，把全选勾选值赋给每个人
        sckList[i].checked = this.checked;
      }

    }
```

```
// 3. 小多选功能 -> 全选
for (var j = 0; j < sckList.length; j++) {
  // 3.1 每个小多选框 - 点击事件
  sckList[j].onclick = function () {
    // 3.2 声明一个综合状态变量
    var isAll;
    // 循环 遍历 小多选框，判断
    for (k = 0; k < sckList.length; k++) {
      // 3.3 有一个未选中的，isAll就是false，代表全选的多选框不能选中
      if (sckList[k].checked == false) {
        isAll = false;
      }
    }

    // 3.4 把综合得出的结果，赋予给全选的多选框
    all.checked = (isAll == false) ? false : true;
    // 如果有人被记(小多选框未选中)了，多选框就得 是false，如果没人 被记录，则多选框为true
  }
}

// 4. 实现+功能
for (var i = 0; i < addList.length; i++) {
  // 4.1 每个+按钮 的 绑定索引
  addList[i].ind = i;
  // 4.2 每个+按钮 的 点击事件
  addList[i].onclick = function () {
    // 4.3 (核心): 通过索引对应找到标签
    // 4.4 this代表+按钮，a标签，用索引找到对应的数字输入框
    var numInput = numInputList[this.ind];
    // 4.5 用索引找到对应-按钮
    var reduceBtn = reduceList[this.ind];
    // 4.6 给数字+1赋予回去
    numInput.value = ++numInput.value;
    // 4.7 减号标签，移除禁用样式
    reduceBtn.classList.remove("disabled");
    // 4.8 用索引找到对应 单价td标签
    var priceTd = priceTdList[this.ind];
    // 4.9 用索引找到对应 小计td标签
    var totalTd = totalTdList[this.ind];
    // 4.10 把单价*数量，等于小计
    totalTd.innerHTML = parseInt(priceTd.innerHTML) * numInput.value + "¥";
  }
}

// 5. 实现-功能
for (var j = 0; j < reduceList.length; j++) {
  // 5.1 每个-按钮 的 索引
  reduceList[j].ind = j;
  // 5.2 每个-按钮 的 点击事件
  reduceList[j].onclick = function () {
    // 5.3 获取索引对应的数量输入框

    var numInput = numInputList[this.ind];
```

```
        // 5.4 值大于1，才能进入减1代码
        if (numInput.value > 1) {
          // 5.5 把当前值取出-1赋予回去
          numInput.value = --numInput.value;
          // 5.6 用索引找到对应 单价td标签
          var priceTd = priceTdList[this.ind];
          // 5.7 用索引找到对应 小计td标签
          var totalTd = totalTdList[this.ind];
          // 5.8 把小计单独计算
          totalTd.innerHTML = parseInt(priceTd.innerHTML) * numInput.value + "¥";
        }

        // 5.9 商品数量改完后，判断如果是1，开启-按钮的禁用样式(但是功能是靠上面的if判断的-因为只有表单标签才能禁用按钮的交互-无法触发点击事件)
        if (numInput.value == 1) {
          this.classList.add("disabled");
        }
      }
    }
```

Day03

课上练习

8.2 利用标签关系-找标签

练习1: 有3组div>p, 嵌套关系的标签, 请在点击p标签时, 打印对应的div标签. 而且要打印div的标签名字, (请用标签关系来寻找, 不使用索引对应)

标签模板

```
<body>
    <div title="1">
        <p class="my_p">我是p1</p>
    </div>
    <div title="2">
        <p class="my_p">我是p2</p>
    </div>
    <div title="3">
        <p class="my_p">我是p3</p>
    </div>
    <script>


    </script>
</body>
```

正确代码JS

```
// 练习1: 有3组div>p，嵌套关系的标签，请在点击p标签时，打印对应的div标签. 而且要打印div的标签名字，(请
```

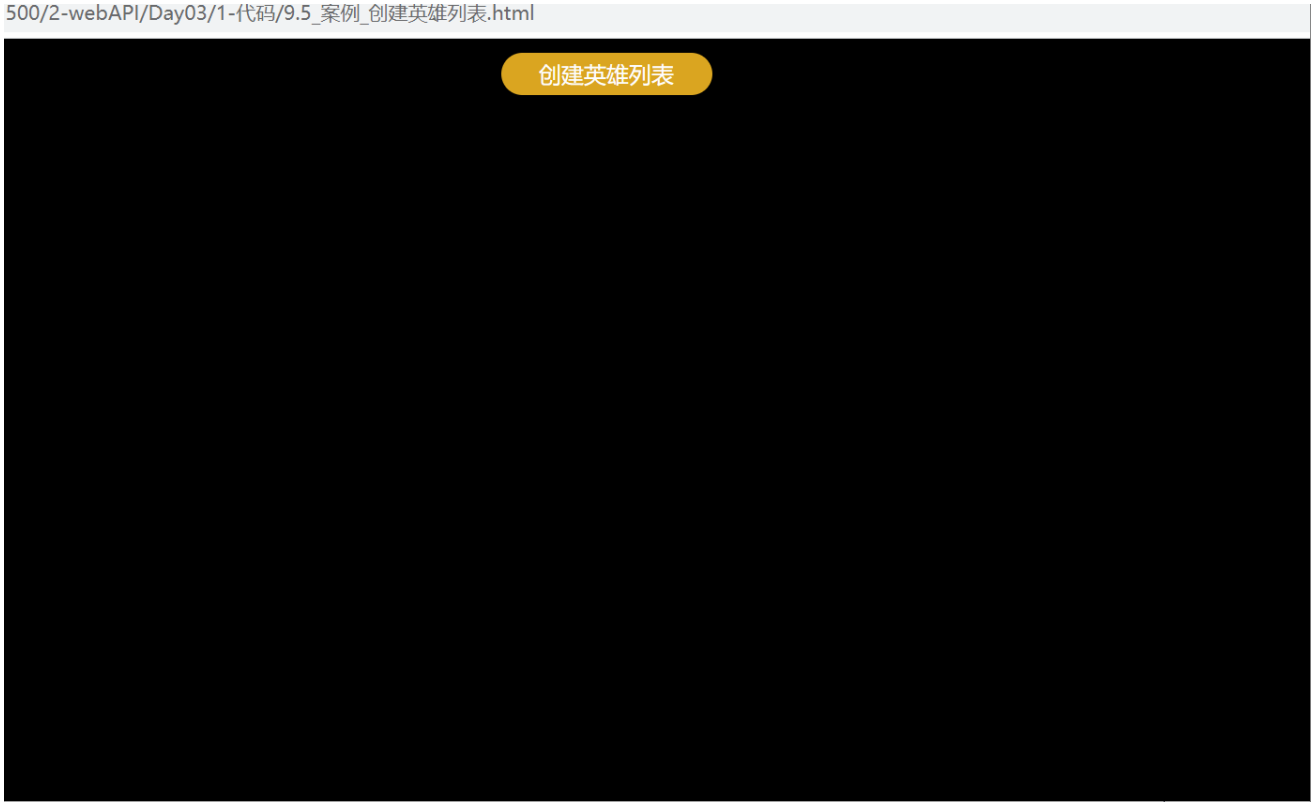
```
用标签关系来寻找，不使用索引对应）
// 1. 标签该获取就获取
var pList = document.querySelectorAll(".my_p"); // [p, p, p]
// 2. 每个p - 点击事件
for (var i = 0; i < pList.length; i++) {
    pList[i].onclick = function(){
        // 3. 通过看标签的关系，发现调用p的parentNode即可
        console.log(this.parentNode);
        console.log(this.parentNode.nodeName);
    }
}

//总结:

// 1. 获取标签，还是可以使用querySelectorAll等方法

// 2. 只有标签之间有父子 / 兄弟关系 才能用上面的属性来获取\
```

案例

9.5 案例 - 英雄列表

素材在images/素材-看序号

500/2-webAPI/Day03/1-代码/9.5_案例_创建英雄列表.html



标签和样式可复制

```
<!DOCTYPE html>
<html lang="en">

    <head>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>创建英雄列表</title>
    <style>
        * {
            margin: 0;
            padding: 0;
        }

        ul {
            list-style: none;
        }

        a {
            width: 150px;
            height: 30px;
            text-decoration: none;
            display: block;
            margin: 10px auto;
            background: goldenrod;
            color: #fff;
            border-radius: 15px;
            line-height: 30px;
        }

        a:hover {
            background: gold;
            color: #666;
        }

        body {
            background: #000;
            text-align: center;
        }

        .list {
            width: 500px;
            margin: 20px auto;
        }

        .list li {
            float: left;
            position: relative;
            transition: all 1s;
            margin-top: 15px;
        }

        .list li:first-child {
            margin-left: 0;
        }


        .list li:hover {
```

```
                transform: scale(1.3);
                z-index: 999;
            }

            .list li img {
                width: 100px;
            }
        </style>
    </head>

    <body>
        <a href="javascript:" id="create">创建英雄列表</a>
        <ul id="list" class="list">
        </ul>
        <script>
            var dataArr = [
                { name: '司马懿', imgSrc: '01.jpg' },
                { name: '女娲', imgSrc: '02.jpg' },
                { name: '百里守约', imgSrc: '03.jpg' },
                { name: '亚瑟', imgSrc: '04.jpg' },
                { name: '虞姬', imgSrc: '05.jpg' },
                { name: '张良', imgSrc: '06.jpg' },
                { name: '安其拉', imgSrc: '07.jpg' },
                { name: '李白', imgSrc: '08.jpg' },
                { name: '阿珂', imgSrc: '09.jpg' },
                { name: '墨子', imgSrc: '10.jpg' },
                { name: '鲁班', imgSrc: '11.jpg' },
                { name: '嬴政', imgSrc: '12.jpg' },
                { name: '孙膑', imgSrc: '13.jpg' },
                { name: '周瑜', imgSrc: '14.jpg' },
                { name: 'XXX', imgSrc: '15.jpg' },
                { name: 'XXX', imgSrc: '16.jpg' },
                { name: 'XXX', imgSrc: '17.jpg' },
                { name: 'XXX', imgSrc: '18.jpg' },
                { name: 'XXX', imgSrc: '19.jpg' },
                { name: 'XXX', imgSrc: '20.jpg' }
            ];
        </script>
    </body>

</html>
```

正确答案: js

```
// 需求：点击a标签，根据JS中准备好的数据，创建英雄头像列表
// 思路：a标签点击事件触发函数执行时，遍历数组里的每个对象，核心是遍历数组里的每个对象时，要给每个对象创建出对应的标签容器来承载数据，然后把标签添加到页面上显示

// 1. 准备好要铺设到页面上的数据
var dataArr = [
    { name: '司马懿', imgSrc: '01.jpg' },

    { name: '女娲', imgSrc: '02.jpg' },
```

```
    { name: '百里守约', imgSrc: '03.jpg' },
    { name: '亚瑟', imgSrc: '04.jpg' },
    { name: '虞姬', imgSrc: '05.jpg' },
    { name: '张良', imgSrc: '06.jpg' },
    { name: '安其拉', imgSrc: '07.jpg' },
    { name: '李白', imgSrc: '08.jpg' },
    { name: '阿珂', imgSrc: '09.jpg' },
    { name: '墨子', imgSrc: '10.jpg' },
    { name: '鲁班', imgSrc: '11.jpg' },
    { name: '嬴政', imgSrc: '12.jpg' },
    { name: '孙膑', imgSrc: '13.jpg' },
    { name: '周瑜', imgSrc: '14.jpg' },
    { name: 'XXX', imgSrc: '15.jpg' },
    { name: 'XXX', imgSrc: '16.jpg' },
    { name: 'XXX', imgSrc: '17.jpg' },
    { name: 'XXX', imgSrc: '18.jpg' },
    { name: 'XXX', imgSrc: '19.jpg' },
    { name: 'XXX', imgSrc: '20.jpg' }
];
// 2. 获取标签
var theUL = document.getElementById("list");
var theA = document.getElementById("create");

// 3. a标签 - 点击事件
theA.onclick = function () {
    theUL.innerHTML = ""; // 把UL的内容清空掉(始终保持只有20个li)
    // 3.1 产生索引
    for (var i = 0; i < dataArr.length; i++) {
        // 3.2 每次取出一个数据对象
        var heroObj = dataArr[i];
        // 3.3 数据对象 - 创造标签
        var theLi = document.createElement("li");
        var theImg = document.createElement("img");
        // 3.4 把对象上的数据赋予到对应的标签上
        theImg.title = heroObj['name'];
        theImg.src = "./images/9.5/" + heroObj.imgSrc;
        // 3.5 把li添加到ul上，img添加到li上
        theUL.appendChild(theLi);
        theLi.appendChild(theImg);
    }
}

// 注意：for循环 每次取出的数据对象变量和标签 都是新的，但是往页面上进行增量铺设
```

9.6 案例 - 微博发布

素材图片在images/看序号

效果演示:

· 使用事件：oninput, 实时监测输入框内容变化, 不用像onchange必须失去焦点才判断触发

标签和样式模板

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <title>发布微博</title>
        <style>
            * {
                margin: 0;
                padding: 0;
            }

            ul {
                list-style: none;
            }

            .w {
                width: 900px;
                margin: 0 auto;
            }

            .controls textarea {
                width: 878px;
                height: 100px;
                resize: none;
                border-radius: 10px;
                outline: none;

                padding-left: 20px;
```

```
        padding-top: 10px;
        font-size: 18px;
    }

    .controls {
        overflow: hidden;
    }

    .controls div {
        float: right;
    }

    .controls div span {
        color: #666;
    }

    .controls div .useCount {
        color: red;
    }

    .controls div button {
        width: 100px;
        outline: none;
        border: none;
        background: rgb(0, 132, 255);
        height: 30px;
        cursor: pointer;
        color: #fff;
        font: bold 14px '宋体';
        transition: all 0.5s;
    }

    .controls div button:hover {
        background: rgb(0, 225, 255);
    }

    .controls div button:disabled {
        background: rgba(0, 225, 255, 0.5);
    }

    .contentList {
        margin-top: 50px;
    }

    .contentList li {
        padding: 20px 0;
        border-bottom: 1px dashed #ccc;
        position: relative;
    }

    .contentList li .info {
        position: relative;

    }
```

```
            .contentList li .info span {
                position: absolute;
                top: 15px;
                left: 100px;
                font: bold 16px '宋体';
            }

            .contentList li .info p {
                position: absolute;
                top: 40px;
                left: 100px;
                color: #aaa;
                font-size: 12px;
            }

            .contentList img {
                width: 80px;
                border-radius: 50%;
            }

            .contentList li .content {
                padding-left: 100px;
                color: #666;
                word-break: break-all;
            }

            .contentList li .the_del{
                position: absolute;
                right: 0;
                top: 0;
                font-size: 28px;
                cursor: pointer;
            }
        </style>
    </head>
    <body>
        <div class="w">
            <!-- 操作的界面 -->
            <div class="controls">
                <img src="./images/9.6/tip.png" alt=""><br>
                <textarea placeholder="说点什么吧..." id="area" cols="30" rows="10"
maxlength="200"></textarea>
                <div>
                    <span class="useCount" id="useCount">0</span>
                    <span>/</span>
                    <span>200</span>
                    <button id="send">发布</button>
                </div>
            </div>
            <!-- 微博内容列表 -->
            <div class="contentList">

                <ul id="list">
```

```
                </ul>
            </div>
        </div>
    </body>

</html>
```

正确代码js

```
// 需求：用户输入内容，使用文字实时改变，点击发布按钮，把内容展示到下面，注意展示时先后顺序
// 思路：
// 1. 输入框oninput事件，监测内容长度，改变文字数量，maxLength控制了最大输入长度
// 2. 按钮绑定点击事件，创建一套标签赋予属性和内容，添加到相应标签上

// 1. 获取标签
var area = document.getElementById("area");
var useCount = document.getElementById("useCount");
var send = document.getElementById("send");
var myUL = document.getElementById("list");

// 2. 输入框 - oninput事件
area.oninput = function(){
    // 2.1 获取输入框值的长度 - 显示已使用个数
    useCount.innerHTML = this.value.length;
}

// 3. 发布按钮 - 点击事件
send.onclick = function(){
    // 3.1 发布内容长度为0，不允许代码继续执行
    if (area.value.length === 0) {
        alert("请输入微博内容");
        return; // 阻止代码继续向下执行
    }

    // 3.2 创建需要的标签
    var theLi = document.createElement("li");
    var userDiv = document.createElement("div");
    var contentDiv = document.createElement("div");
    var delSpan = document.createElement("span");
    var headImg = document.createElement("img");
    var nameSpan = document.createElement("span");
    var timeP = document.createElement("p");

    // 3.3 设置相关属性
    userDiv.className = "info";
    contentDiv.className = "content";
    delSpan.className = "the_del";
    headImg.src = "./images/9.6/03.jpg";

    // 3.4 设置内容 (value / innerHTML)

    nameSpan.innerHTML = "百里守约";
```

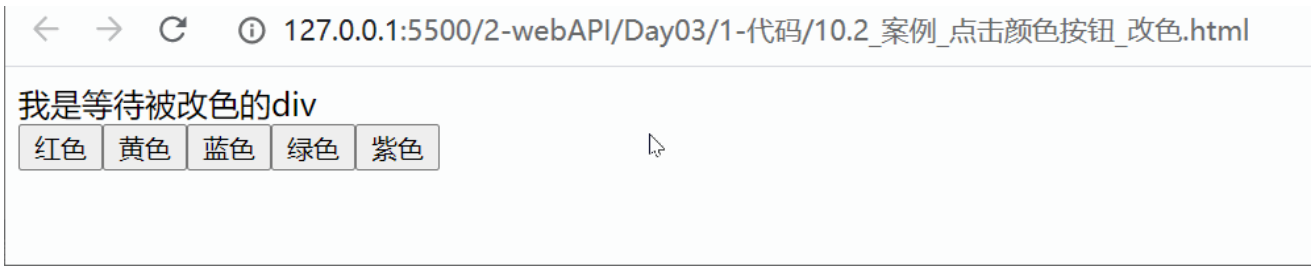
```
    timeP.innerHTML = "发布于" + formatTime(new Date());
    contentDiv.innerHTML = area.value;
    delSpan.innerHTML = "X";

    // 3.5 把相应的标签进行组合
    theLi.appendChild(userDiv);
    theLi.appendChild(contentDiv);
    theLi.appendChild(delSpan);
    userDiv.appendChild(headImg);
    userDiv.appendChild(nameSpan);
    userDiv.appendChild(timeP);

    // 3.6 新增加的li要插入到最前面
    myUL.insertBefore(theLi, myUL.firstElementChild);

    // 3.7 点击发布后，把内容清空，把useCount归为0
    area.value = "";
    useCount.innerHTML = 0;
}

// 4. 定义时间处理函数，把事件格式化成 年-月-日 时:分:秒 格式(要前边补0)
function formatTime(dateObj){
    var year = dateObj.getFullYear();
    var month = dateObj.getMonth() + 1;
    var day = dateObj.getDate();
    var hour = dateObj.getHours();
    var minutes = dateObj.getMinutes();
    var seconds = dateObj.getSeconds();

    month = month < 10 ? '0' + month : month;
    day = day < 10 ? '0' + day : day;
    hour = hour < 10 ? '0' + hour : hour;
    minutes = minutes < 10 ? '0' + minutes : minutes;
    seconds = seconds < 10 ? '0' + seconds : seconds;

    return year + "年" + month + "月" + day + "日" + " " + [hour, minutes, seconds].join(":");
}

// 总结：表单标签value，普通标签夹着的内容innerHTML
```

9.7 案例 - 微博删除

基于上面代码, 实现删除功能


```
// 5. X标签 - 点击事件
delSpan.onclick = function(){
   // 5.1 要删除整个li
   myUL.removeChild(theLi);
}
```

10.2 案例_点击颜色按钮变色




```
// 需求：点击对应文字的按钮，设置div的背景色为相应的颜色
// 思路：准备文字数组和颜色英文字母数组，一定要索引对应，用文字数组创建按钮，给按钮绑定索引，点击按钮获取索引，然后找到索引对应的颜色英文，赋予给标签上

// 1. 准备数据
var colorStrArr = ["红色", "黄色", "蓝色", "绿色", "紫色"];
var letterStrArr = ["red", "yellow", "blue", "green", "purple"];

// 2. 获取标签
var targetDiv = document.getElementById("targetDiv");
var buttonWrap = document.getElementById("buttonWrap");

// 3. 创建按钮
// 3.1 产生索引

for (var i = 0; i < colorStrArr.length; i++) {
```

```
    // 3.2 每次从数组里取出一个中文 colorStrArr[i]
    // 3.3 创建button按钮，给予内容添加到页面上
    var theBtn = document.createElement("button");
    theBtn.innerHTML = colorStrArr[i];
    buttonWrap.appendChild(theBtn);
}

// 4. 按钮绑定事件
// 4.1 (注意要在创建之后获取) - 获取所有button
var buttonList = document.querySelectorAll("#buttonWrap>button");
// 4.2 产生索引
for (var j = 0; j < buttonList.length; j++) {
    // 4.3 每个按钮 - 绑定索引
    buttonList[j].ind = j;
    // 4.4 每个按钮 - 点击事件
    buttonList[j].onclick = function(){
        // 4.5 点击，获取索引 - this.ind
        // 4.6 英文数组中取出英文字母颜色 - 赋予到标签div
        targetDiv.style.backgroundColor = letterStrArr[this.ind];
    }
}
```

作业

作业1 - 就业龙虎榜

效果演示:

标签和样式模板:

```
<!DOCTYPE html>
```

```html
<html lang="en">

    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <title>就业龙虎榜</title>
        <style>
            * {
                margin: 0;
                padding: 0;
            }

            body {
                background: rgb(2, 78, 15);
            }

            a {
                text-decoration: none;
                color: #aaa;
            }

            h1 {
                text-align: center;
                color: #fff;
                margin: 20px 0;

            }

            table {
                margin: 0 auto;
                width: 800px;
                border-collapse: collapse;
            }

            th {
                padding: 10px;
                background: purple;
                color: gold;
                font-size: 20px;
            }

            td,
            th {
                border: 1px solid #ccc;
            }

            td {
                padding: 10px;
                color: #666;
                text-align: center;
                font-size: 16px;

            }
```

```
            tbody tr {
                background: #fff;
            }

            tbody tr:hover {
                background: gold;
            }
        </style>
    </head>

    <body>
        <h1>就业榜</h1>
        <div id="box1">

        </div>
        <script>


            // 后端的数据1
            var heads1 = ['姓名', '年龄', '性别', '学号', '薪资', '城市', '操作'];
            var datas1 = [
                { name: '欧阳霸天', age: 19, gender: '男', stuId: '1001', salary: '20000', city:
'上海' },
                { name: '令狐霸天', age: 19, gender: '男', stuId: '1001', salary: '20000', city:
'北京' },
                { name: '诸葛霸天', age: 19, gender: '男', stuId: '1001', salary: '20000', city:
'南京' },
                { name: '欧阳炸炸', age: 19, gender: '男', stuId: '1001', salary: '20000', city:
'南京' },
                { name: '欧阳炸炸', age: 19, gender: '男', stuId: '1001', salary: '20000', city:
'南京' },
                { name: '欧阳炸炸1', age: 19, gender: '男', stuId: '1001', salary: '20000', city:
'南京' },
                { name: '欧阳炸炸2', age: 19, gender: '男', stuId: '1001', salary: '20000', city:
'南京' },
                { name: '欧阳炸炸3', age: 19, gender: '男', stuId: '1001', salary: '20000', city:
'南京' },
                { name: '欧阳炸炸4', age: 19, gender: '男', stuId: '1001', salary: '20000', city:
'南京' }

            ];



        </script>
    </body>

</html>
```

正确答案js代码:

```
/*
```

```
  功能：根据数据创建表格
  参数：
    element  元素   创建的表格放入的元素
    heads    数组   表头中数据
    datas    数据   表体中的数据
*/
function table(element, heads, datas) {

    // 2. 创建table元素追加到div
    var table = document.createElement('table');
    element.appendChild(table);
    // 3.【创建thead元素】追加到table
    var thead = document.createElement('thead');
    table.appendChild(thead);
    // 3.1 创建一个tr追加到thead
    var headTr = document.createElement('tr');
    thead.appendChild(headTr);
    // 3.2 循环遍历heads数据，取出一个数据就创建一个th追加到tr中
    for (var i = 0; i < heads.length; i++) {
        var th = document.createElement('th');
        // 设置th的内容
        th.innerText = heads[i];
        headTr.appendChild(th);
    }

    // 4.【创建tbody元素】追加到table中
    var tbody = document.createElement('tbody');
    table.appendChild(tbody);
    // 4.1 循环遍历datas数据，取出一个对象，就创建一个tr追加到tbody中
    for (var i = 0; i < datas.length; i++) {
        var bodyTr = document.createElement('tr');
        tbody.appendChild(bodyTr);
        // 取出一个对象
        var obj = datas[i];
        // 4.2 用for-in循环遍历取出的对象中的属性，每取出要给属性值就创建一个td追加tr
        for (var key in obj) {
            var td = document.createElement('td');
            bodyTr.appendChild(td);
            td.innerText = obj[key];
        }
        // 4.3 创建最后一个td，设置内容为删除，追加到tr中
        var lastTd = document.createElement('td');
        bodyTr.appendChild(lastTd);
        lastTd.innerHTML = '<a href="javascript:">删除</a>';
    }

};

// 获取元素
var box1 = document.querySelector('#box1');
// 调用生成一个表格放到box1中
table(box1, heads1, datas1);
```

```
// 删除
// 1. 获取一组删除按钮
var dels = document.querySelectorAll('a');
// 2. 注册点击事件
for (var i = 0; i < dels.length; i++) {
    dels[i].onclick = function () {
        // 3. 提示是否删除
        var isOk = confirm('您真的要删吗? 可要想好了');
        // 4. 若是, 找当前删除按钮所在的tr, 找tbody
        if (isOk) {
            var tr = this.parentNode.parentNode;
            var tbody = this.parentNode.parentNode.parentNode;
            tbody.removeChild(tr);
        }
    };
}
```

Day04

课上练习

12.7 点击弹出提示

练习1. 导航div中, 左右分别有2个按钮(个人信息和退出网页按钮) - 3个标签点击分别弹出提示


```
<style>
    #wrap{
        background-color: red;
        overflow: hidden;
    }
    #per{
        background-color: blue;
        float: left;
    }
    #quit{
        background-color: yellow;
        float: right;
    }
</style>
</head>
<body>
    <div id="wrap">
        <span id="per">个人信息</span>
        <span id="quit">退出?</span>
```

```
    </div>
    <script>
        // 需求：父级div下，左侧点击弹出个人提示，导航上点击弹出我是导航，退出上点击提示是否退出
        // 思路：获取3个标签，分别绑定点击事件，点击显示对应弹窗

        // 1. 获取标签
        var wrap = document.getElementById("wrap");
        var per = document.getElementById("per");
        var quit = document.getElementById("quit");

        // 2. 分别绑定点击事件
        wrap.onclick = function(){
            alert("我是导航");
        }
        per.onclick = function(ev){
            ev.stopPropagation();
            alert("个人信息");
        }
        quit.onclick = function(ev){
            ev.stopPropagation();
            confirm("退出");
        }
    </script>
</body>
```

13.2 课上练习

练习1: 实现组合键 ctrl + 回车 实现弹窗

```
<body>
    <input type="text" id="user">
    <script>
        // 需求：当用户输入完名字以后，按下键盘ctrl+enter，弹出用户输入的内容
        // 思路：给输入框绑定键盘按下事件，判断ctrlKey的状态值 和这次按下的keyCode值

        // 1. 获取标签
        var user = document.getElementById("user");
        // 2. 绑定键盘按下事件
        user.onkeydown = function(ev){ // 先按住ctrl键，再按下enter时，触发事件执行
            // 2.1 获取ctrlKey的状态值，和当前按下键的keyCode值，判断(回车是13)
            if (ev.ctrlKey && ev.keyCode === 13){
                // 2.2 符合判断弹窗
                alert(this.value);
            }
        }
    </script>
</body>
```

案例

12.3 案例 - 移动瞄准

标签和样式模板:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <title>移动瞄准</title>
    <style type="text/css">
        div {
            height: 300px;
            background: black;
            color: white;
            cursor: crosshair;
        }
    </style>
</head>

<body>
    <div id="myDiv">坐标x, y</div>
</body>

</html>
```

正确代码js:

```
// 需求：鼠标在黑色区域内移动，在左上角显示坐标点
// 思路：给黑色div绑定鼠标移动事件，获取鼠标坐标点

// 1. 获取标签
var myDiv = document.getElementById('myDiv');
// 2. 添加事件
myDiv.onmousemove = function (ev) {
   // 3. 获取坐标，显示到对应标签上
   this.innerHTML = "坐标x:" + ev.clientX + " 坐标y:" + ev.clientY;
}
```

12.4 案例 - 小鸟跟随 - 鼠标移动

素材在images/看序号里

效果演示:

标签和样式模板:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>鼠标控制小鸟移动</title>
  <style>
    img {
      width: 100px;
      border:1px solid red;
      position: absolute;
    }
  </style>
</head>
<body>
  <img src="./images/xn.png" alt="" id="myImg">
</body>
</html>
```

正确的js代码

```
// 需求：鼠标移动时，让小鸟图片跟着鼠标移动
// 思路：给网页绑定鼠标移动事件，获取坐标x，y，设置图片的left和top影响图片的移动

// 1. 获取标签
var myImg = document.getElementById("myImg");
// 2. 绑定鼠标移动事件
document.onmousemove = function(ev){
    // 3. 把鼠标当前的坐标 赋予给标签的相关属性
    // myImg.style.left = ev.clientX + "px";
    // myImg.style.top = ev.clientY + "px";

    // 4. 让鼠标中心点在图片中心点

    myImg.style.left = ev.clientX - myImg.offsetWidth / 2 + "px";
```

```
    myImg.style.top = ev.clientY - myImg.offsetHeight / 2 + "px";
}
```

13.3 案例 - 键盘控制小鸟移动


```
// 需求：按下键盘4个方向键，控制小鸟进行移动
// 思路：给网页绑定键盘onkeydown事件，然后获取用户按下的keyCode键盘码，在图片当前的位置
offsetLeft/offsetTop位置基础上，根据方向判断，应该是增加/减少对应方向的值

// 1. 获取图片元素
var myImg = document.getElementById('myImg');

// 2. 给页面绑定键盘按下事件
document.onkeydown = function(ev){
   // 3. 获取到用户点击的4个方向键 (37 38 39 40 左上右下)
   // 4. 一个keyCode和4种值对比，选择了switch/case写法
   switch (ev.keyCode) {
       case 37:
           // 当用户按的左，则在现有位置 - 10，设置回去，影响标签位置
           myImg.style.left = myImg.offsetLeft - 10 + "px";
           // 同时修改图片，绕着垂直Y轴，旋转-180度(头从右转到左)
           myImg.className = "toLeft";
           break;
       case 38:
           myImg.style.top = myImg.offsetTop - 10 + "px";
           break;
       case 39:
           myImg.style.left = myImg.offsetLeft + 10 + "px";
           myImg.className = "";
           break;
       case 40:
           myImg.style.top = myImg.offsetTop + 10 + "px";
           break;
  }
}
```

14.1 案例 - 给新增li绑定事件

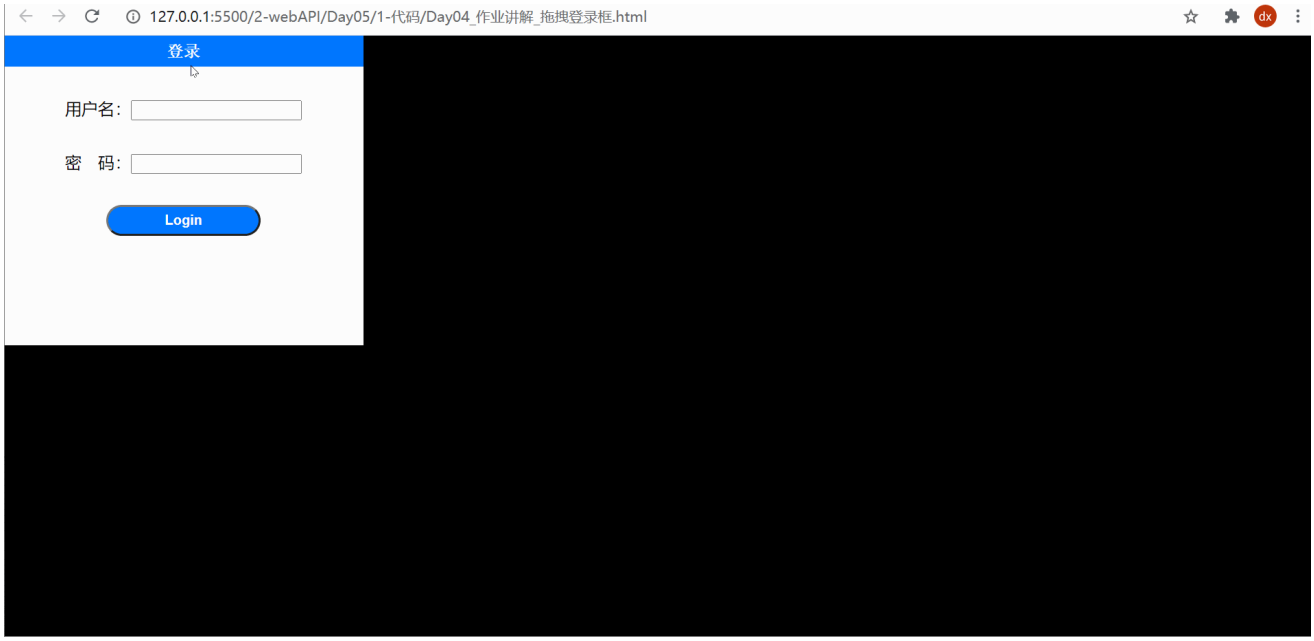
> 页面现有li, 点击弹出自己内容, 点击按钮可以在尾部新增li按钮, 新增的li点击也可以弹出li里的内容

```
// 需求：页面现有li，点击弹出自己内容，点击按钮可以在尾部新增li按钮，新增的li点击也可以弹出li里的内容
// 思路：给父级ul绑定点击事件，通过ev.target.nodeName判断是否是LI标签，是的话再去弹出内容

// 之前的循环做法（用于和新方式对比）
// // 1. 获取标签
// var btn = document.getElementById("btn");
// var myUL = document.getElementById("myUL");
// // 2. 新增按钮 - 点击事件
// btn.onclick = function(){
//     // 2.1 新建li - 设置内容
//     var theLi = document.createElement("li");
//     theLi.innerHTML = "我是新的li";
//     myUL.appendChild(theLi);

//     // 2.2 新建li - 点击事件
//     theLi.onclick = function(){
//         console.log(this);
//     }
// }

// // 3. 获取页面现有li
// var liList = document.querySelectorAll("#myUL>li");
// // 4. 产生索引
// for (var i = 0; i < liList.length; i++) {
//     // 5. 给每个li绑定点击事件
//     liList[i].onclick = function(){
//         console.log(this);
//     }
// }


// 优化后，使用事件委托
myUL.onclick = function(ev) {
   if (ev.target.nodeName == "LI") {
       console.log(ev.target);
  }
}
```

作业

作业1 - 鼠标按下拖拽蓝色区域 - 移动窗口

鼠标抬起后, 窗口就不再跟随鼠标移动咯, 实现案例_拖拽功能效果



标签和样式模板:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <title>Document</title>
        <link rel="stylesheet" href="css/drag.css">

    </head>
    <body>
        <div class="login">
            <div class="tip">登录</div>
            <div>
                <p>
                    <label>
                        用户名: <input type="text">
                    </label>
                </p>
                <p>
                    <label>
                        密 码: <input type="text">
                    </label>
                </p>
                <p>
                    <button>Login</button>
                </p>
            </div>
        </div>
        <!-- <script src="js/drag.js"></script> -->
        <script type="text/javascript">

        </script>
```

```
    </body>
</html>
```

正确答案js代码

```
// 鼠标按下在登录div，元素可以移动【移动事件，按下有事件级处理程序，鼠标在文档移动】
// 鼠标抬起在登录div，元素不可以移动
// 鼠标移动的时候，大div跟着移动
// 获取元素
var tip = document.querySelector('.tip');
var login = document.querySelector('.login');

// 添加事件
tip.onmousedown = function (e1) {
    var x1 = e1.offsetX;
    var y1 = e1.offsetY;

    // 移动
    document.onmousemove = function (e) {

        // 获取x和y
        var x = e.clientX;
        var y = e.clientY;

        // 因为元素左上角对齐，所以一移动，元素就会自动左上角对齐鼠标
        // 我们只要再让元移动回去既可
        // 我们需要知道鼠标按下时，距离这个元素的x和y的位置

        // 设置给login
        login.style.left = x - x1 + 'px';
        login.style.top = y - y1 + 'px';


    }

}

tip.onmouseup = function () {

    document.onmousemove = null;

}
```

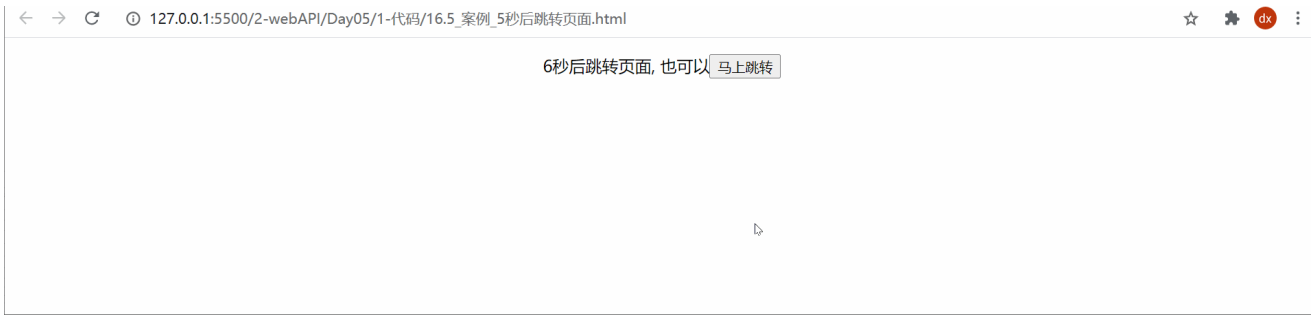
Day05

课上练习

暂无

案例

16.5 案例 - 5秒后跳转页面



标签和样式模板:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>案例_5秒后跳转页面</title>
    <style>
        .my_p{
            text-align: center;
        }
    </style>
</head>
<body>
    <p class="my_p">
        <span id="time"></span>秒后跳转页面，也可以<button id="btn">马上跳转</button>
    </p>
</body>
</html>
```

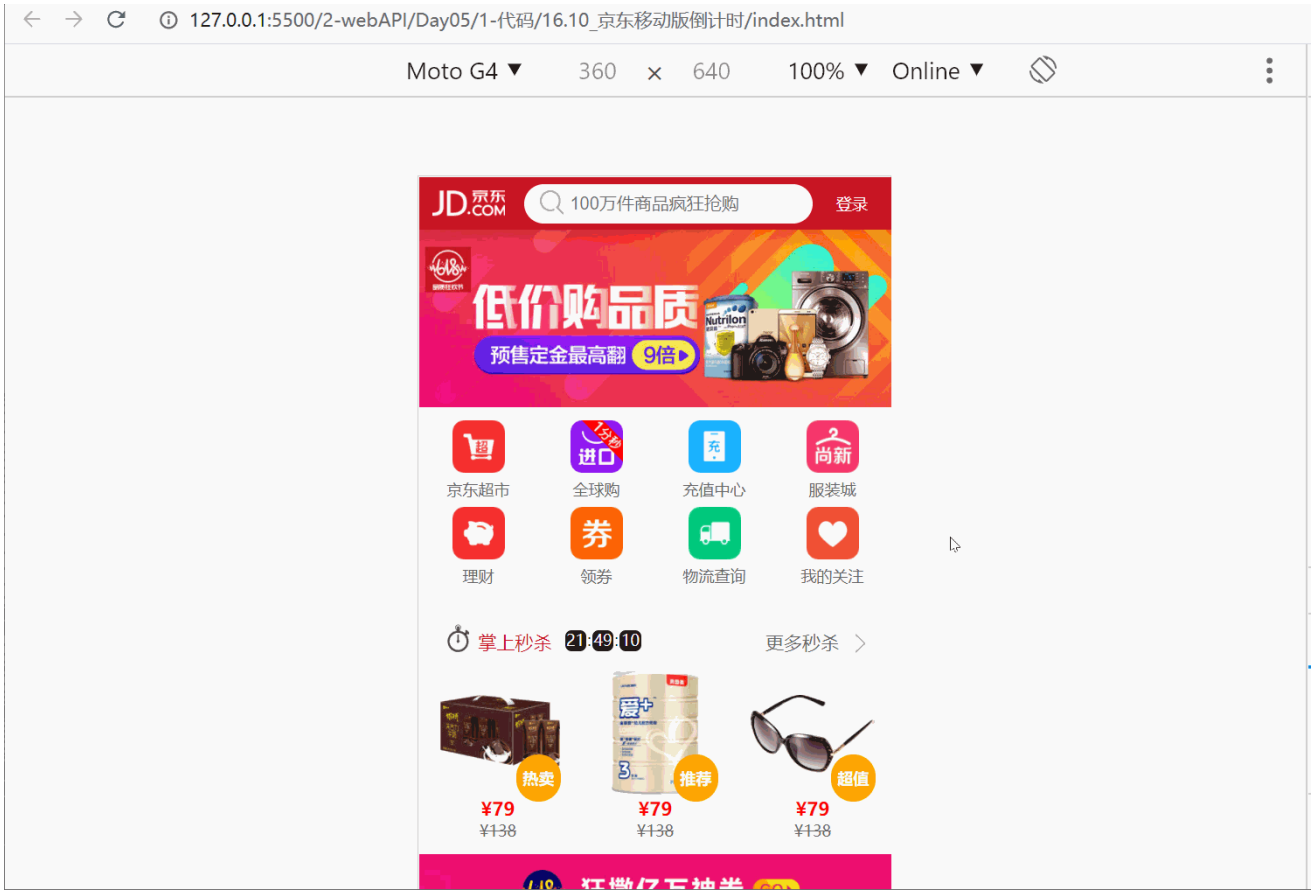
正确的js代码:

```
// 需求：打开网页，5秒后，跳转到指定的页面，也可以点击按钮马上跳转
// 思路：
// 1. 创建计时器，然后判断数字到0，消除计时器，跳转页面
// 2. 按钮绑定点击事件，直接点击跳转

// 1. 获取标签
var theTime = document.getElementById("time");
var btn = document.getElementById("btn");

// 2. 按钮 - 点击事件
btn.onclick = function(){
    window.location.href = "http://www.baidu.com";
}


// 3. 倒计时时间
```

```
// 3.1 网页刚打开 - 先赋予到页面一份
var i = 5;
theTime.innerHTML = i;
// 3.2 计时器
var t = setInterval(function(){
    // 3.3 倒计时结束
    if (i == 0) {
        // 3.4 消除计时器
        clearInterval(t);
        // 3.5 跳转页面
        window.location.href = "http://www.baidu.com";
    } else {
        // 3.6 正常-1，更新到页面
        theTime.innerHTML = --i;
    }
}, 1000);
```

16.10 案例 - 京东移动版倒计时

文件模板在images/下看序号



如果时间到0了，应该提示用户alert("活动结束"); // 显示00:00:00即可，当然活动结束时间，自己可以在代码里传入时间

```
// 需求：秒杀中，做一个倒计时效果，当为0时，弹窗活动结束
// 思路：

// 1. 定义活动结束时间对象变量，当前系统时间的时间变量，分别获取毫秒然后得到减法的差值
```

```
// 2. 把差值转换成时分秒格式，设置到页面上
// 3. 创建计时器，每秒钟都执行上面的代码一次（让页面动起来）

// 1. 获取标签
var hoursLi = document.getElementById("hours");
var minutesLi = document.getElementById("minutes");
var secondsLi = document.getElementById("seconds");

// 2. 定义函数，改变倒计时标签内容
function timer(){
    // 2.1 设置活动结束时间，和系统现在时间
    var nowDate = new Date();
    var endDate = new Date("2020/08/20 15:14:11"); // 注意格式
    // 2.2 得到差值秒，然后计算出时分秒
    var theS = (endDate.getTime() - nowDate.getTime()) / 1000;
    var h = Math.floor(theS / 60 / 60);
    var m = Math.floor(theS / 60 % 60);
    var s = Math.floor(theS % 60);

    // 2.3 把时分秒赋予到标签上
    hoursLi.innerHTML = (h < 10 ? '0' + h : h);
    minutesLi.innerHTML = (m < 10 ? '0' + m : m);
    secondsLi.innerHTML = (s < 10 ? '0' + s : s);

    // 2.4 如果剩下的秒<=0了，停止计时器，弹出提示，标签显示0
    if (theS <= 0) {
        clearInterval(t);
        hoursLi.innerHTML = "00";
        minutesLi.innerHTML = "00";
        secondsLi.innerHTML = "00";
        alert("活动结束");
    }

}

// 3. 网页打开时，调用一次(防止空屏)，然后每秒调用一次函数执行
var t = setInterval(function () {
    timer();
}, 1000);
timer();
```

17.3 案例 - 电脑开机

素材在images/下看序号

标签和样式模板:

```
<!DOCTYPE html>
<html>

    <head lang="en">
        <meta charset="UTF-8">
        <title>案例_开机效果</title>
        <style>
            .box {
                width: 322px;
                position: fixed;
                bottom: 0;
                right: 0;
                overflow: hidden;
            }

            span {
                position: absolute;
                top: 0;
                right: 0;
                width: 30px;

                height: 20px;
```

```
                cursor: pointer;
            }

            .bd {
                height: 105px;
                transition: all 600ms;
            }

            .box {
                transition: all 600ms;
            }
        </style>
    </head>

    <body>
        <div class="box" id="box">
            <span id="closeButton"></span>
            <div class="hd" id="headPart">
                <img src="images/17.2/t.jpg" alt="" />
            </div>
            <div class="bd" id="bottomPart">
                <img src="images/17.2/b.jpg" alt="" />
            </div>
        </div>
        <script>

        </script>
    </body>


</html>
```

正确的js代码

```
// 需求：点击x按钮，底部图片高度切成0，然后整个容器宽度切成0
// 思路：
// 1. 给x按钮绑定点击事件，点击后，给底部图片设置高度为0
// 2. 因为是使用css过渡动画，所以给底部图片绑定transitionend事件
// 3. 当事件触发时，把整个容器宽度位置为0

// 1.获取元素
var closeButton = document.getElementById('closeButton');
var bottomPart = document.getElementById('bottomPart');
var box = document.getElementById('box');

// 2. x按钮的点击事件
closeButton.onclick = function (ev) {
    // 2.1 把底部图片高度设为0
    bottomPart.style.height = 0;
}


// 3. 监听底部图片过度动画结束
```

```
bottomPart.addEventListener("transitionend", function () {
    // 3.1 把整个容器宽度设为0
    box.style.width = 0;
})
```

17.4 案例 - 轮播图

演示色块页面 - 看原理

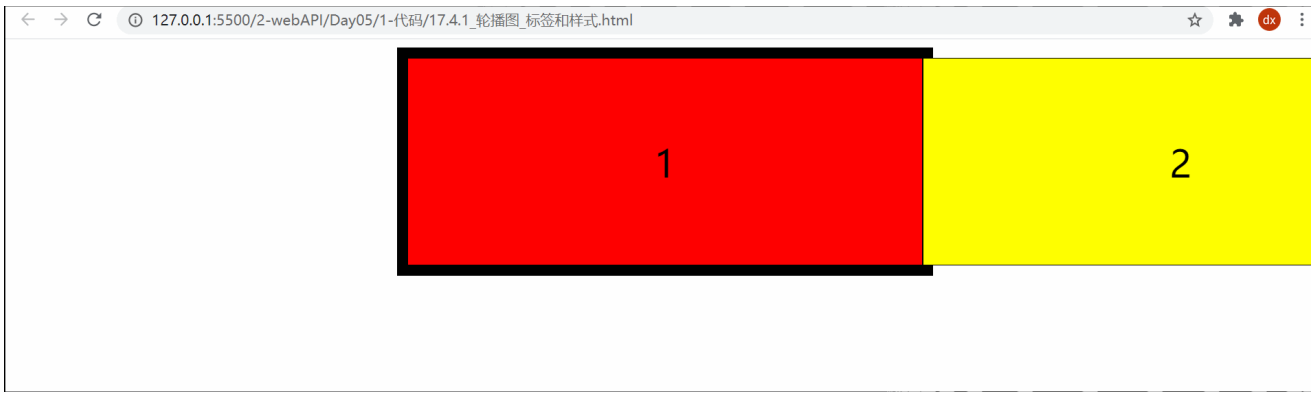
再参考原型图: https://www.processon.com/view/link/5f3d2b287d9c0806d4225515

标签模板和样式

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>轮播图_标签和样式</title>
    <style>
        #clientDiv{
            width: 500px;
            height: 200px;
            margin: 0 auto;
            border: 10px  solid black;
            position: relative;
        }
        #content{
            height: 200px;
            position: absolute;
            left: 0;
            top: 0;
        }
        #content>div{
            float: left;
            width: 500px;
            height: 200px;
            font-size: 36px;
            line-height: 200px;
            text-align: center;
            border: 1px solid black;
            color: black;
            box-sizing: border-box;
        }
        .one{
            background-color: red;
        }
        .two{
            background-color: yellow;
        }

        .three{
```

```
            background-color: blue;
        }
        .four{
            background-color: green;
        }
        .five{
            background-color: purple;
        }
    </style>
</head>
<body>
    <!-- 1.1 标签结构，可视区域>内容容器>内容 -->
    <div id="clientDiv">
        <div id="content">
            <div class="one">1</div>
            <div class="two">2</div>
            <div class="three">3</div>
            <div class="four">4</div>
            <div class="five">5</div>
        </div>
    </div>
</body>
</html>
```

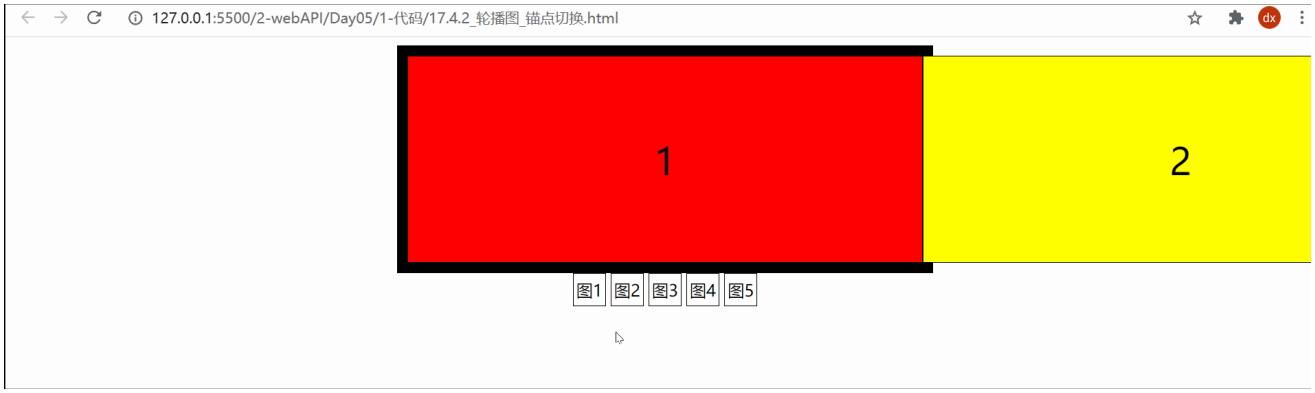
17.4.1 - 轮播图 - 标签和样式



父级宽度要动态计算, js代码

```
// 1.2 知道css样式
// clientDiv -> 目的是可视区域
// content -> 宽度是所有项目宽度之和，内容容器
// 改变content的left，达到露出来的目的

// 1.3 (重要) 父级容器宽度，需要动态计算
var contentDiv = document.getElementById("content");
var itemWidth = contentDiv.children[0].offsetWidth; // 每个图片宽度(因为都一样所以随便取个下角标0)
contentDiv.style.width = contentDiv.children.length * itemWidth + "px";上
```

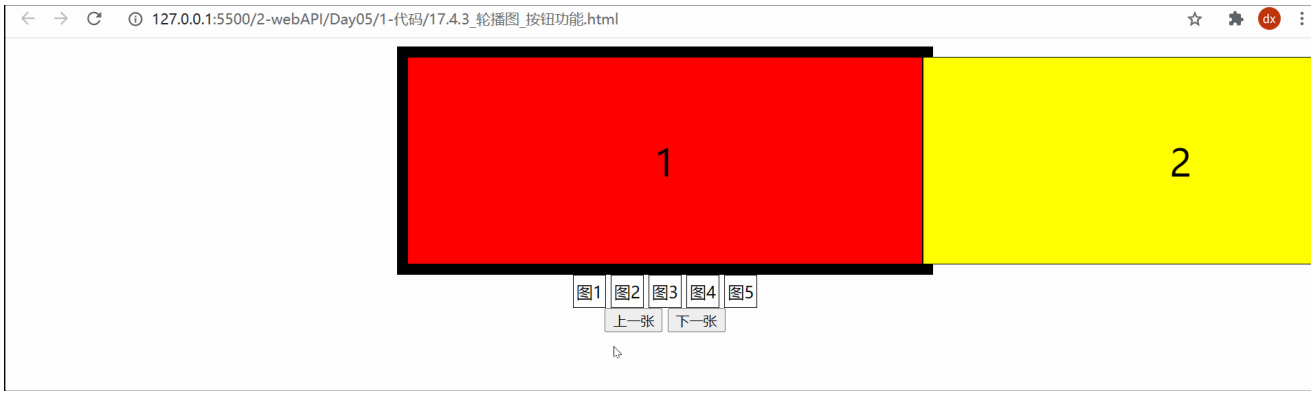
17.4.2 - 轮播图 - 锚点点击切换图片



正确代码

```
// 2.2 循环锚点，分别绑定点击事件
var spanList = document.querySelectorAll("#spotWrap>span");
for (var i = 0; i < spanList.length; i++) {
    spanList[i].onclick = function(ev){
        // 2.3 获取自定义属性上的索引 - 点击span的索引
        var index = this.getAttribute("ind");
        // 2.4 通过索引*每个图片容器宽度，修改contentDiv的宽度，漏出想看的
        contentDiv.style.left = - index * itemWidth + "px";
    }
}
```

17.4.3 - 轮播图 - 按钮(上一张, 下一张)

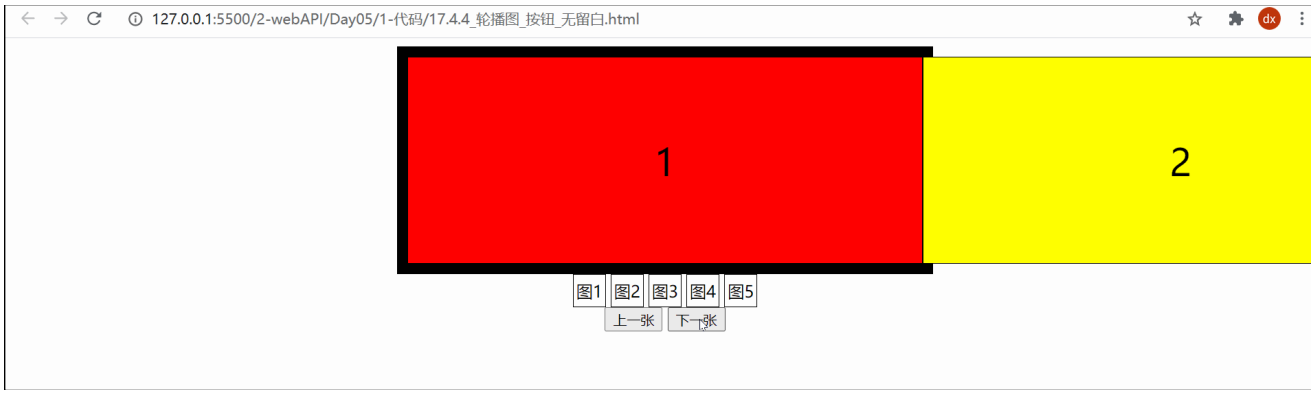


正确代码:

```
// 3.2 获取标签
var lastBtn = document.getElementById("last");
var nextBtn = document.getElementById("next");

// 3.3 下一张
nextBtn.onclick = function(ev){
    ind++; // 索引+1
    contentDiv.style.left = - ind * itemWidth + "px"; // 使用公式，算出contentDiv应该移动的-left值
}
// 3.5 上一张
lastBtn.onclick = function(ev){
```

```
    ind--; // 索引-1
    contentDiv.style.left = - ind * itemWidth + "px"; // 使用公式，算出contentDiv应该移动的-left值
}
```

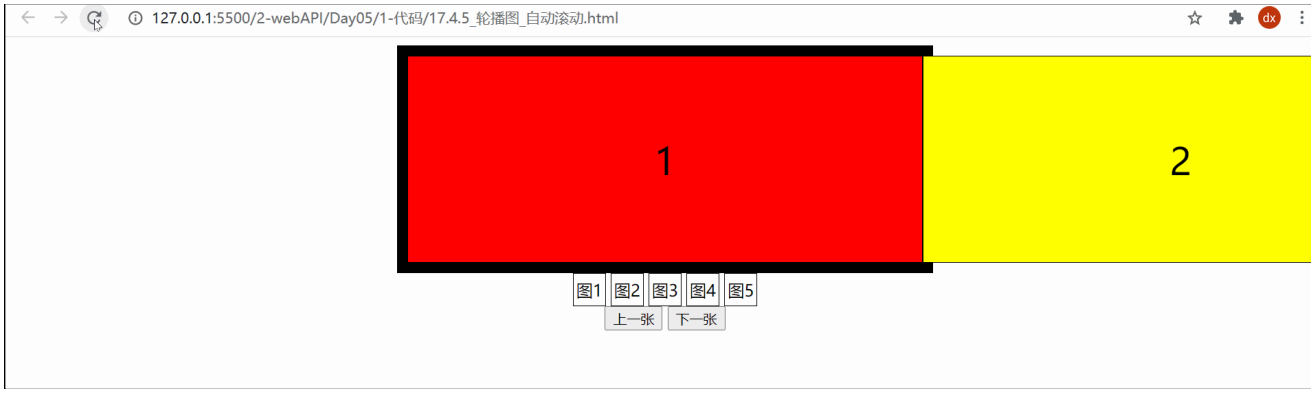
17.4.4 - 轮播图 - 按钮(循环滚动)



正确代码:

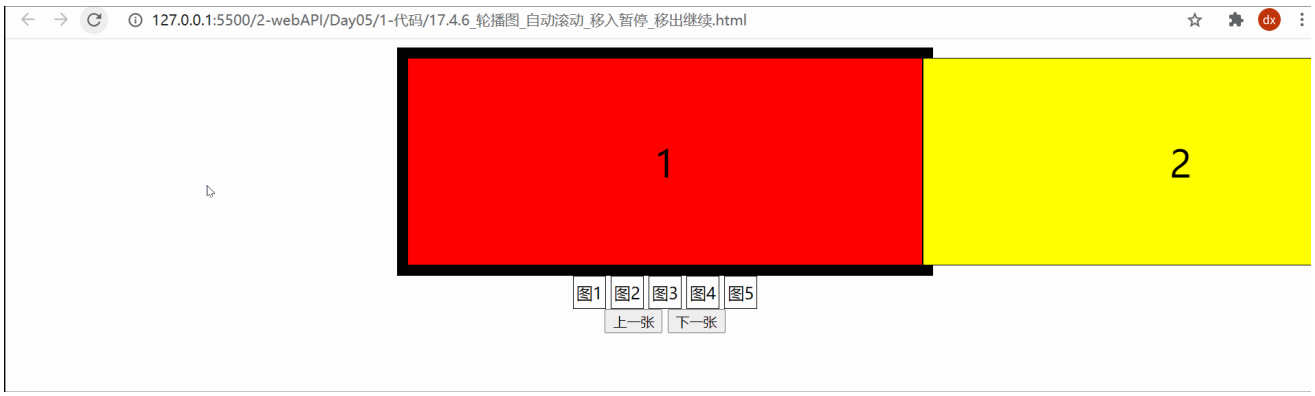
```
var ind = 0;
var lastBtn = document.getElementById("last");
var nextBtn = document.getElementById("next");

nextBtn.onclick = function(ev){
    // 4.1 点击下一张时，如果当前索引已经是最后一张了，就把index变成-1，让下面ind++后代表索引0
    if (ind == contentDiv.children.length - 1) {
        ind = -1;
    }
    ind++;
    contentDiv.style.left = - ind * itemWidth + "px";
}

lastBtn.onclick = function(ev){
    // 4.2 点击上一张时，如果当前索引为0，就把index变成子元素个数长度，让下面ind--后代表最后一个图对应的索引值
    if (ind === 0) {
        ind = contentDiv.children.length;
    }
    ind--;
    contentDiv.style.left = - ind * itemWidth + "px";
}
```

17.4.5 - 轮播图 - 自动滚动




```
// 5.1 创建计时器
var t = setInterval(function(){
    // 5.2 和点击下一张按钮的逻辑是相同的
    if (ind == contentDiv.children.length - 1) {
        ind = -1;
    }
    ind++;
    // 5.3 使用公式进行位移
    contentDiv.style.left = - ind * itemWidth + "px";
}, 3000);
```

17.4.6 - 轮播图 - 自动滚动 - 移入停止, 移出继续




```
// 6.0 改装成函数
var t;
function intervalMove() {
    t = setInterval(function () {
        if (ind == contentDiv.children.length - 1) {
            ind = -1;
        }
        ind++;
        contentDiv.style.left = - ind * itemWidth + "px";
    }, 3000);
}
intervalMove();


// 6.1 获取可视标签
```

```
var clientDiv = document.getElementById("clientDiv");
// 6.2 移入暂停计时器
clientDiv.onmouseenter = function (ev) {
    clearInterval(t);
}
// 6.3 移出开启计时器
clientDiv.onmouseleave = function (ev) {
    intervalMove();
}
```

17.4.7 - 轮播图 - 自动滚动和点击冲突解决

锚点 和按钮发生点击时 - 停止计时器 - 位移后 - 再把计时器创建


```
var clientDiv = document.getElementById("clientDiv");
clientDiv.onmouseenter = function(){
    clearInterval(t);
}
clientDiv.onmouseout = function(){
    intervalMove();
}
```

作业

素材images/17.5 品优购 轮播图-模板 (自行删除js代码发给学生)

额外练习

放大镜

素材在images/看文件夹名字

效果如下:

标签和样式准备

```
<!DOCTYPE html>
<html>
<head lang="en">
```

```
    <meta charset="UTF-8">
    <title>案例_放大镜</title>
    <style>
        * {
            margin: 0;
            padding: 0;
        }

        .w {
            width: 1190px;
            margin: 0 auto;

        }

        .fdj {
            margin-top: 20px;
        }

        .fdj .leftBox {
            width: 400px;
            height: 400px;
            border: 1px solid green;
            float: left;
            position: relative;
        }

        .fdj .tool {
            width: 250px;
            height: 250px;
            background: gold;
            opacity: .5;
            filter: alpha(opacity=50);
            position: absolute;
            top: 0;
            left: 0;
            cursor: move;
            /* 默认隐藏 */
            display: none;

        }

        /* 给小黄加上active 就会显示 */
        .fdj .tool.active {
            display: block;
        }

        .fdj .rightBox {
            width: 500px;
            height: 500px;
            border: 1px solid yellow;
            float: left;
            overflow: hidden;

            /* 隐藏 */
```

```
            display: none;
            position: relative;
        }

        /* 加上active表示显示 */
        .fdj .rightBox.active {
            display: block;
        }

        .fdj .rightBox img {
            position: absolute;
        }
    </style>
    <style>
        body{
            background-color: black;
        }
    </style>
</head>

<body>
    <div class="w">
        <div class="fdj">
            <div class="leftBox" id="_leftBox">
                <!-- 小图 -->
                <img src="./images/15.3/m.jpg" alt="" />
                <!-- 小黄盒子 -->
                <div class="tool" id="_tool"></div>
            </div>
            <div class="rightBox" id="_rightBox">
                <img id="_bImg" src="./images/15.3/b.jpg" alt="" />
            </div>
        </div>
    </div>
    <script>

    </script>
</body>

</html>
```

正确的js代码

```
// 需求：鼠标移入左侧盒子出现放大器，在左侧小图上移动，让右侧大图显示出对应区域的图片内容
// 原理：
// 1. 左边是小图，右边是2倍的大图（图片本身尺寸，前提）
// 2. 右侧(rightBox)，设置overflow:hidden; 超出区域隐藏掉，只露出该看到的图片部分即可
// 思路：鼠标控制左边黄色放大器 -> 影响 -> 右侧大图的位移

// 1. 获取标签
var leftBox = document.getElementById("_leftBox");

var leftTool = document.getElementById("_tool"); // 放大器
```

```
var rightBox = document.getElementById("_rightBox");
var bigImg = document.getElementById("_bImg");

// 2. 先实现左侧黄色放大器功能
// 2.1 左侧盒子 - 移入事件
leftBox.onmouseover = function(ev){
    // 2.2 放大器 - 出现
    leftTool.style.display = "block";
    // 2.3 右侧盒子 - 出现
    rightBox.style.display = "block";
}
// 2.4 左侧盒子 - 移出事件
leftBox.onmouseout = function(ev){
    // 2.5 放大器 - 隐藏
    leftTool.style.display = "none";
    // 2.6 右侧盒子 - 隐藏
    rightBox.style.display = "none";
}


// 3. 在左侧盒子上绑定鼠标移动事件
leftBox.onmousemove = function(ev){
    // 3.1 获取鼠标坐标点 - 修改放大器 - 位置
    // 3.2 clientX/clientY相对浏览器窗口 - leftBox左上多的值offsetLeft和offsetTop，才是放大器 真正的位置
    // 3.3 分别减去放大器宽高的一半，为了让放大器 中心点和鼠标重合
    var theLeft = ev.clientX - leftBox.offsetLeft - (leftTool.offsetWidth / 2);
    var theTop = ev.clientY - leftBox.offsetTop - (leftTool.offsetHeight / 2);
    // 3.4 不要让放大器超出左侧盒子范围 - 判断临界值
    // 3.5 如果放大器的left 要小于0，强制改成0
    if (theLeft < 0) {
        theLeft = 0;
        // 3.6 如果放大器的left 大于 左侧盒子容器宽度 - 放大器宽度，则强制设置为左侧大盒子宽度-放大器宽度 (难点-画图)
    } else if (theLeft > leftBox.offsetWidth - leftTool.offsetWidth){
        theLeft = leftBox.offsetWidth - leftTool.offsetWidth;
    }

    // 3.7 与上同理，设置垂直方向临界值
    if (theTop < 0) {
        theTop = 0;
    } else if (theTop > leftBox.offsetHeight - leftTool.offsetHeight){
        theTop = leftBox.offsetHeight - leftTool.offsetHeight;
    }

    // 3.8 把算好的left值和top值，赋予给放大器
    leftTool.style.left = theLeft + "px";
    leftTool.style.top = theTop + "px";

    // 3.9 同步 影响 右侧img标签的left和top
    // 因为右侧的图片本身 就是左侧图片的2倍大小，所以移动的时候放大2倍
    // 放大器想看手机，右侧大图 就得 往左边移动(负)

    bigImg.style.left = - theLeft * 2 + "px";
```

```
    bigImg.style.top = - theTop * 2 + "px";
}
```

Day06

课上练习

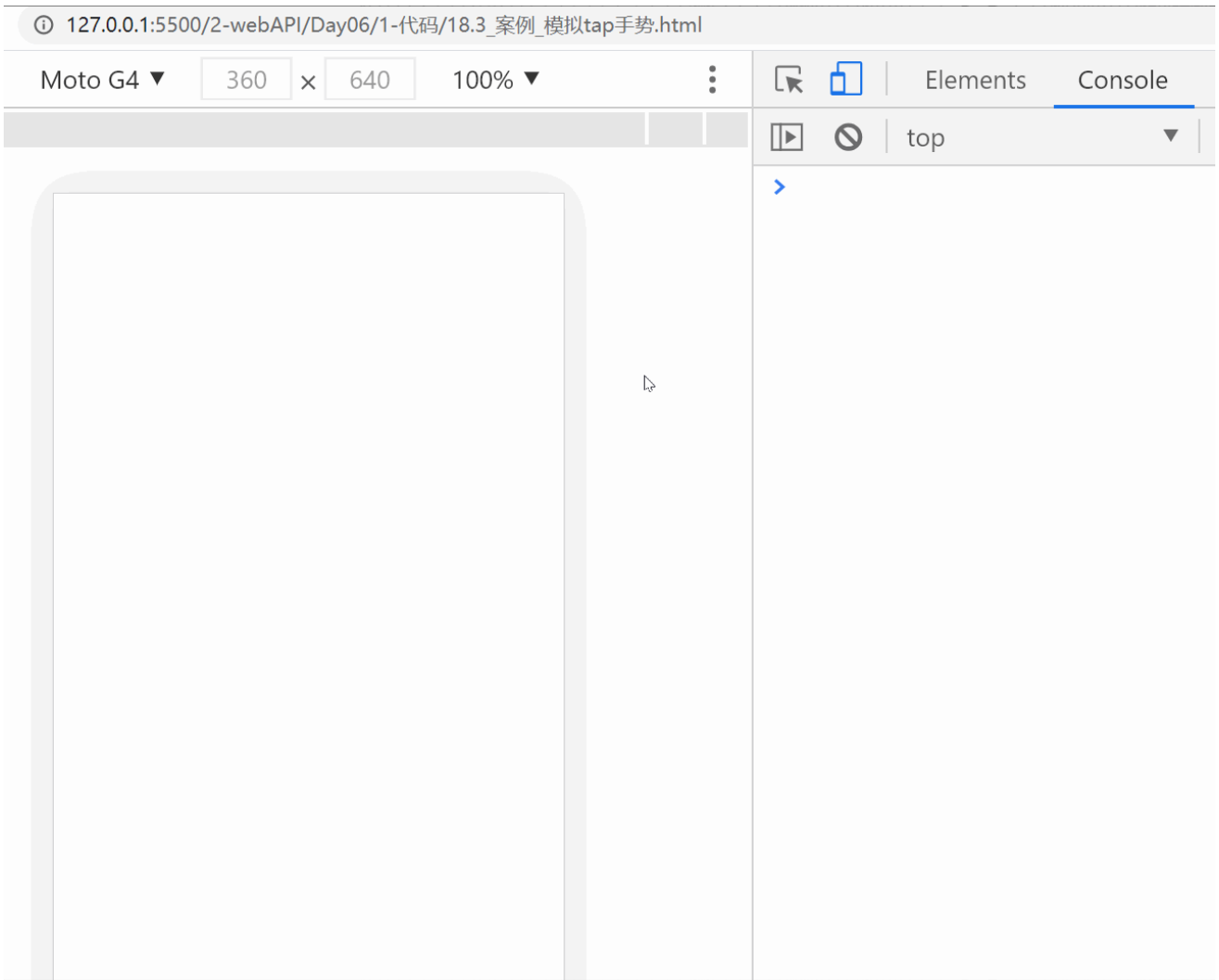
暂无

案例

18.3 案例 - 模拟tap手势

常见的手势

注意：以下手势的实现是在touch事件基础上。我们不需要自己实现以下所有手势，有专门的第三方可以直接拿来使用。以下手势仅仅是了解

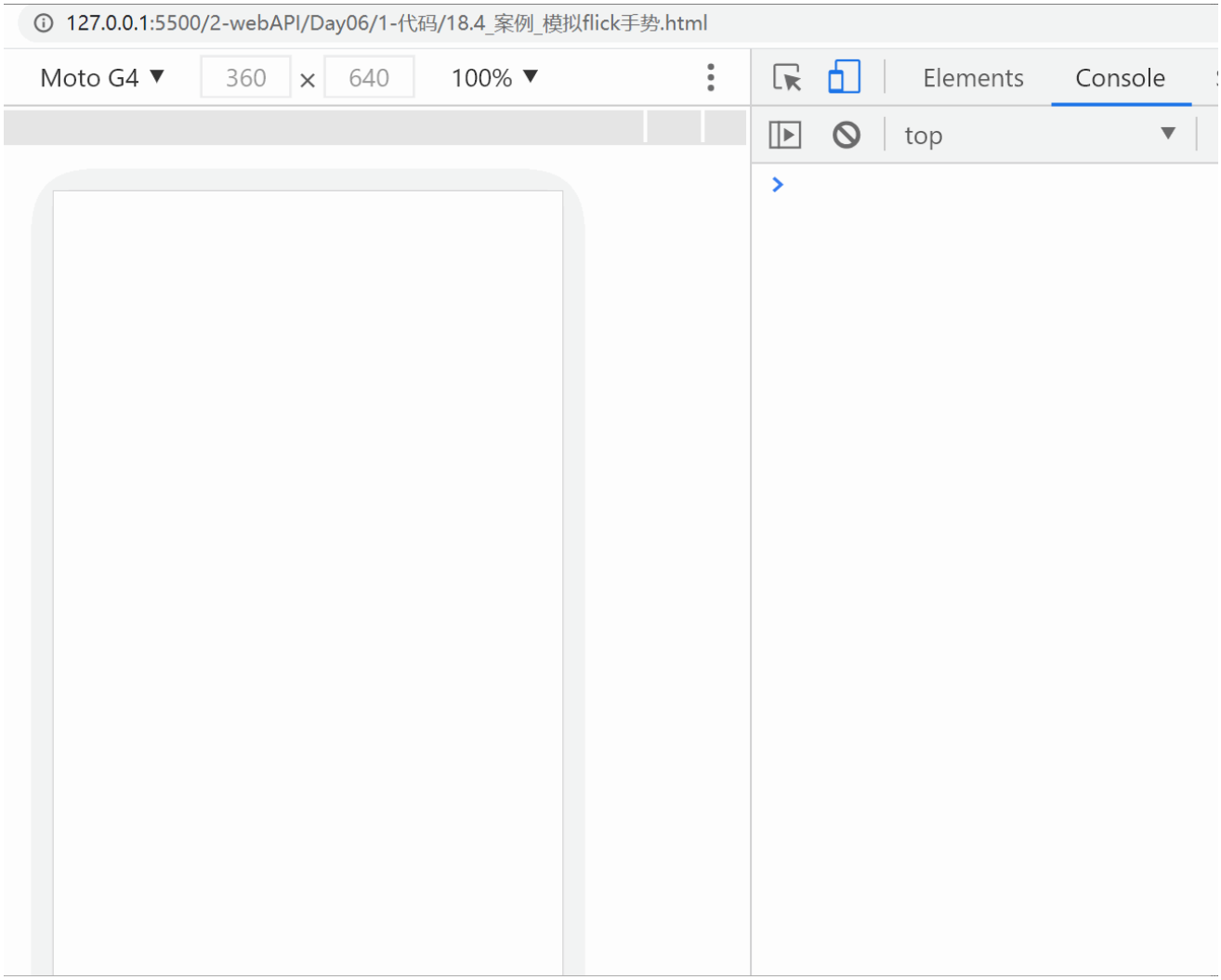



```
// 需求：模拟tap手势 - 移动端点击效果(一按下一抬起，touchstart按下就触发了)
// 思路：
```

```
// 1. 按下时，手指点的x,y坐标
// 2. 抬起时，手指点的x,y坐标
// 3. 判断x和y坐标是同一个点就是tap手势

// 1. 按下时的坐标点x，y
var startX, startY;
document.addEventListener("touchstart", function(ev){
    startX = ev.touches[0].clientX;
    startY = ev.touches[0].clientY;
})

// 2. 抬起时的坐标点x，y
document.addEventListener("touchend",  function(ev){
    var endX = ev.changedTouches[0].clientX;
    var endY = ev.changedTouches[0].clientY;
    // 3. 判断按下时的x，y和抬起时的x，y是否是同一个坐标
    if (startX === endX && startY === endY) {
        console.log("tap事件");
    } else {
        console.log("轻滑事件");
    }
})

// 总结：通过JS原生的touch事件，模拟实现了监测用户- tap手势
```

18.4 案例 - 模拟flick手势

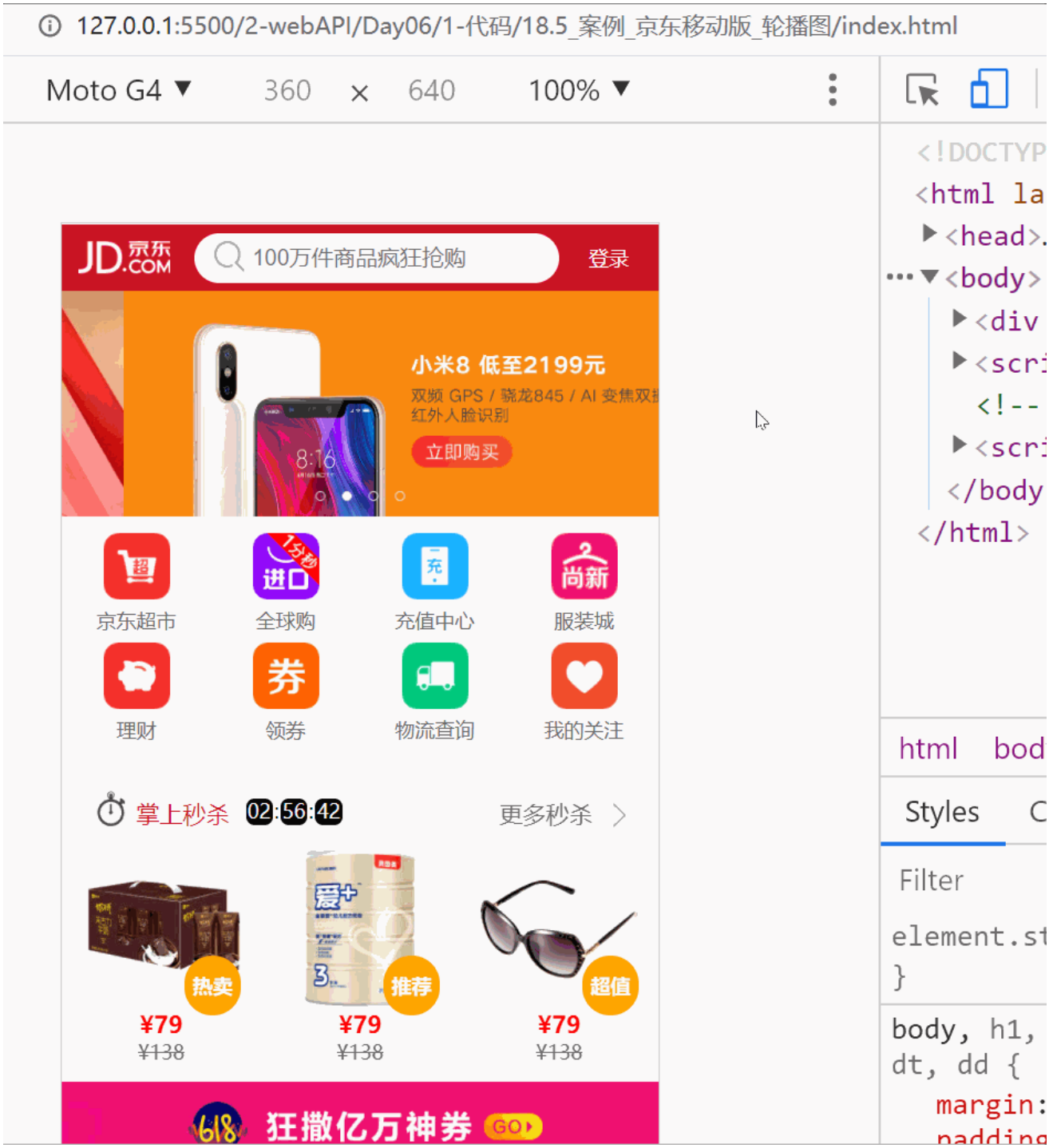
模拟flick手势（轻滑）- 手指按住滑动后抬起




```
// 需求：编写代码 - 监测用户的滑动手势
// 思路：
// 1. 按下时，手指x坐标
// 2. 抬起时，手指x坐标
// 3. 判断抬起时和按下时的x坐标，就能判断出用户是左滑，还是右滑

// 1. 按下时，x坐标
var startX;
document.addEventListener("touchstart", function(ev){
    startX = ev.touches[0].clientX;
})
// 2. 抬起时，x坐标
document.addEventListener("touchend",  function(ev){
    var endX = ev.changedTouches[0].clientX;
    // 3. 判断离开时的x坐标和按下时x坐标的大小
    if (endX > startX) {
        console.log("右侧滑动");
    } else {
        console.log("左侧滑动");
    }
})

// 总结：原生touch事件判断 比较按下和离开时的x坐标点，得出左滑或优化结论
```

18.5 案例 - 移动端京东轮播图

静态项目在素材images/下序号




```
// 需求：根据手指的左滑和右滑来控制轮播图的移动，以及自动滚动的效果，注意使用transform来进行位移

// 思路：
// 1. 判断手指滑动方向，决定index索引的加减，修改图片容器位移的left
// 2. 自动计时器-控制index，影响容器位移


// 1. 获取标签
```

```
var bannerDiv = document.querySelector(".banner");
var contentDiv = document.querySelector(".content");
var itemWidth = contentDiv.children[0].offsetWidth;
var spotList = document.querySelectorAll("#spotWrap>li");
var index = 0; //图片当前下标

// 2. 触摸滚动
var startX = 0;
// 2.1 判断触摸滚动方向
document.addEventListener("touchstart", function (ev) {
    clearInterval(t);
    startX = ev.touches[0].clientX;
})
document.addEventListener("touchend", function (ev) {
    var endX = ev.changedTouches[0].clientX;
    if (endX > startX) { // 右滑
        // 2.2 设置index
        index > 0 && index--;
    } else if (endX < startX) { // 左滑
        index < contentDiv.children.length - 1 && index++;
    }
    // 2.3 设置滚动位置
    contentDiv.style.transform = "translateX(" + -index * itemWidth + "px)";
    // 2.4 同步圆点高亮
    for (var i = 0; i < spotList.length; i++) {
        spotList[i].className = "";
    }
    spotList[index].className = "active";
    go();
})

// 3. 设置自动滚动计时器
var t;
function go() {
    t = setInterval(function () {
        index++; // 修改索引
        if (index === contentDiv.children.length) { // 临界值
            index = 0;
        }
        contentDiv.style.transform = "translateX(" + -index * itemWidth + "px)"; // 位移
        for (var i = 0; i < spotList.length; i++) { // 修改锚点高亮
            spotList[i].className = "";
        }
        spotList[index].className = "active";
    }, 2000);
}
go();
```

19.2 案例 - zepto.js 重构-京东移动端 - 轮播图

效果同上 - 就是代码不同

```
// 需求: 使用zepto.js插件, 重新实现移动端轮播图
```

```
// 思路：引入zepto和zepto.touch.js，使用封装好的方法和事件，优化我们JS基础语法编写的代码，感受效率提高

// 1. 引入zepto.js和zepto.touch.js(注意先后顺序)
// 2. 图片当前下标
var index = 0;
// 3. 监听向左滑动
$(document).on("swipeLeft", function (ev) {
    // 3.1 清除计时器
    clearInterval(t);
    // 3.2 判断临界值
    index == $(".content>.item").length - 1 && (index = -1);
    // 3.3 索引+
    index++;
    // 3.4 通过索引算出容器移动距离，移动，算出哪个分页器高亮
    moveAndLight();
    // 3.5 开启计时器 - 自动轮播
    intervalFn();
})
// 4. 监听向右滑动
$(document).on("swipeRight", function (ev) {
    // 4.1 消除计时器
    clearInterval(t);
    // 4.2 判断临界值
    index == 0 && (index = $(".content>.item").length);
    // 4.3 索引--
    index--;
    // 4.4 移动容器，分页器高亮
    moveAndLight();
    // 4.5 开启计时器 - 自动轮播
    intervalFn();
})

// 负责移动div和分页器高亮
function moveAndLight() {
    $(".content").css({
        transform: "translateX(" + -index * $(".item").width() + "px)"
    })
    $("#spotWrap>li").removeClass("active").eq(index).addClass("active");
}

// 自动轮播的计时器
var t;
function intervalFn() {
    t = setInterval(function () {
        index == $(".content>.item").length - 1 && (index = -1);
        index++;
        moveAndLight();
    }, 2000);
}
intervalFn();
```

20.1 案例 - swiper.js - 重构 - 京东移动版 - 轮播图

效果同上

```
// 需求：使用swiper.js插件，重构京东移动端首页的轮播图
// 思路：引入swiper.css和swiper.js，然后准备好标签结构和添加对应的class类名，然后再JS中实例化swiper对象，传入需要的配置对象就ok了

var mySwiper = new Swiper('.swiper-container', {
    autoplay: {
        delay: 3000,//3秒切换一次
    },
    loop: true, // 循环模式选项
    pagination: {
        el: '.swiper-pagination',
    }
})
```

22.1 案例 - 微博信息本地化保存

效果和以前相同 - 就是刷新网页数据还在了

```
// 需求：在原有的发布微博基础上，实现发布时把数据保存在本地，刷新/下次打开可以铺设已有的微博数据
// 思路：
// 1. 获取localStorage对象里的微博信息，如果有JSON.parse()转成数组，如果没有则给予一个空数组
// 2. 在发布时，把数据组织成对象，插入到数组中
// 3. 然后在把数组转成JSON字符串，放入到localStorage对象中
// 4. 网页刚打开，如果本地有数据，还要循环每个对象创建对应标签 显示到页面上

// 1. 先实现发布时的保存功能
// 1.1 先判断localStorage里是否有数据，如果有则取出转成数组类型，如果没有则给arr变量一个空数组(保证下面代码正确)
// (1): 如果本地是空的，arr得到一个空数组
// (2): 如果本地有值，取出现有的微博内容数组，转成数组类型
var arr = localStorage.getItem("weiboArr") ? JSON.parse(localStorage.getItem("weiboArr")) : [];

// 1.2 一条微博需要4个信息，所以用一个对象保存这条微博相关信息
var newsObj = {
    name: "百里守约",
    headImgUrl: "./images/9.6/03.jpg",
    sendTime: sendTime,
    content: area.value
}
// 1.3 插入到数组中
arr.push(newsObj);
// 1.4 因为本地localStorage需要存字符串，所以使用JSON.stringify()转成字符串存储
localStorage.setItem("weiboArr", JSON.stringify(arr));

// 2. 网页刚打开，你还需要从localsStorage里取值，添加到页面上
// 2.1 上面已经从localStorage里取值了
// 2.2 循环本地保存的数组中所有微博信息的对象
for (var i = 0; i < arr.length; i++) {
    // 2.3 从数组里取出每个微博对象
    var newsObj = arr[i];
```

```
    // 2.4 创建标签等，把当前微博信息数据展示到页面上去
    var theLi = document.createElement("li");
    var userDiv = document.createElement("div");
    var contentDiv = document.createElement("div");
    var delSpan = document.createElement("span");
    var headImg = document.createElement("img");
    var nameSpan = document.createElement("span");
    var timeP = document.createElement("p");

    userDiv.className = "info";
    contentDiv.className = "content";
    delSpan.className = "the_del";
    headImg.src = newsObj['headImgUrl'];

    nameSpan.innerHTML = newsObj['name'];
    timeP.innerHTML = "发布于" + newsObj['sendTime'];
    contentDiv.innerHTML = newsObj['content'];
    delSpan.innerHTML = "X";

    theLi.appendChild(userDiv);
    theLi.appendChild(contentDiv);
    theLi.appendChild(delSpan);
    userDiv.appendChild(headImg);
    userDiv.appendChild(nameSpan);
    userDiv.appendChild(timeP);

    myUL.insertBefore(theLi, myUL.firstElementChild);
}
```

22.2 案例 - 微信信息 - 删除功能(同步本地化保存)

```
// 需求：在发布时，可以点击X，删除掉对应微博，在网页刚打开创建的微博li，点击x也可以删除对应的微博.
// 思路：操作同一个arr数组，2套不同地方的x事件，无论谁操作都同步更新到本地localStorage中

delSpan.onclick = function () {
    // 内层函数引用外层函数内的变量-是会直接保存在每次内层函数内（闭包）
    // 3.1 发布时，绑定的删除事件
    // 3.2 找到当前对应对象的在arr里的下标
    arr.splice(arr.indexOf(newsObj), 1);
    myUL.removeChild(theLi);
    // 3.3 最新的数组保存到localStorage中
    localStorage.setItem("weiboArr", JSON.stringify(arr));
}
```

注意: 下面循环中没有函数套函数情况, 我们可以自己造一个函数使用

作业

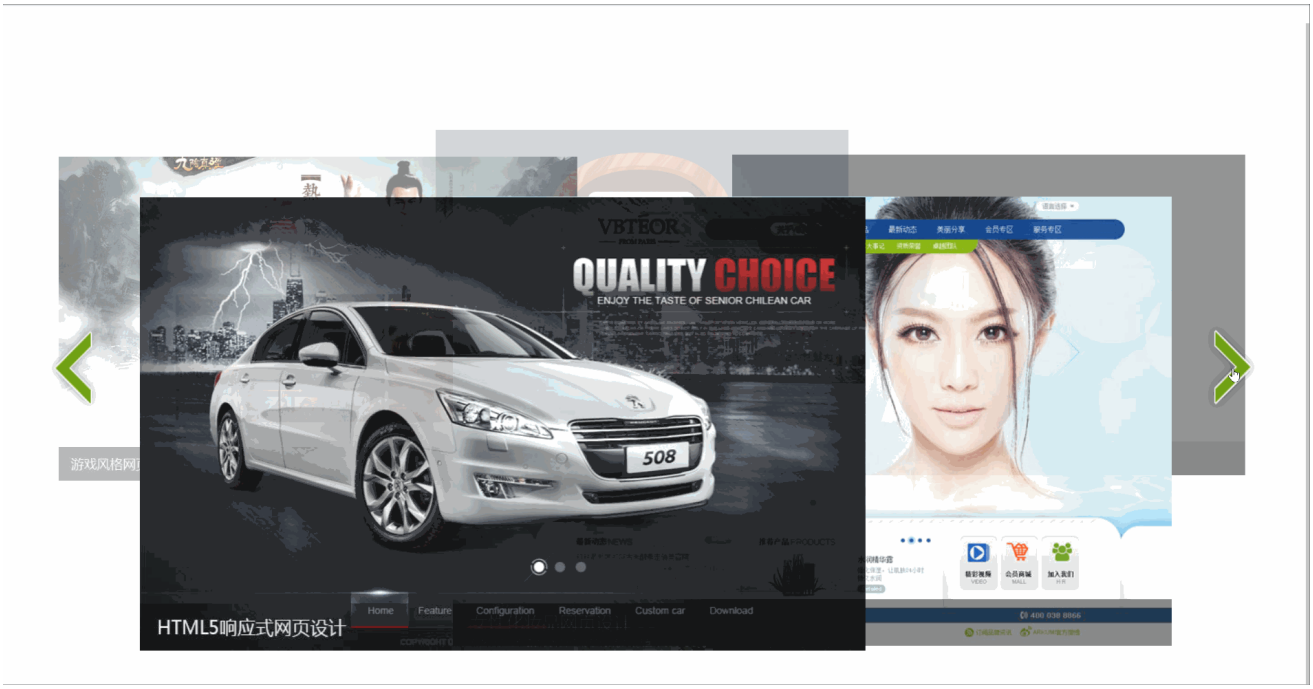
暂无, 把课上案例都搞明白, 再来一遍

额外练习

旋转木马

素材在images/下看文件夹名字

效果演示:



标签和样式准备

```
<!DOCTYPE html>
<html>

<head lang="en">
  <meta charset="UTF-8">
  <title>旋转木马轮播图</title>
  <style>
    * {
      margin: 0;
      padding: 0
    }

    ul {
      list-style: none
    }

    img {
      border: 0;
      vertical-align: top;
    }

    a,
    button {
      cursor: pointer;
    }
```

```css
.wrap {
  width: 1200px;
  margin: 100px auto;
}

.slide {
  height: 500px;
  position: relative;
}

.slide li {
  position: absolute;
  left: 200px;
  top: 0;
  transition: all 350ms;
}

.slide li img {
  width: 100%;
}

.prev,
.next {
  width: 76px;
  height: 112px;
  position: absolute;
  top: 50%;
  margin-top: -56px;
  background: url(./images/20.1/prev.png) no-repeat;
  z-index: 99;
}

.next {
  right: 0;
  background-image: url(./images/20.1/next.png);
}

.slide li.left1 {
  width: 400px;
  top: 20px;
  left: 50px;
  opacity: 0.2;
  z-index: 2;
}

.slide li.left2 {
  width: 600px;
  top: 70px;
  left: 0px;
  opacity: 0.8;
  z-index: 3;
}
```

```
    .slide li.middle {
      width: 800px;
      top: 100px;
      left: 200px;
      opacity: 1;
      z-index: 4;
    }

    .slide li.right2 {
      width: 600px;
      top: 70px;
      left: 600px;
      opacity: 0.8;
      z-index: 3;
    }

    .slide li.right1 {
      width: 400px;
      top: 20px;
      left: 750px;
      opacity: 0.2;
      z-index: 2;
    }
  </style>
</head>

<body>
  <div class="wrap" id="wrap">
    <div class="slide" id="slide">
      <ul>
        <li>
          <a href="#"><img src="images/20.1/slidepic1.jpg" alt="" /></a>
        </li>
        <li>
          <a href="#"><img src="images/20.1/slidepic2.jpg" alt="" /></a>
        </li>
        <li>
          <a href="#"><img src="images/20.1/slidepic3.jpg" alt="" /></a>
        </li>
        <li>
          <a href="#"><img src="images/20.1/slidepic4.jpg" alt="" /></a>
        </li>
        <li>
          <a href="#"><img src="images/20.1/slidepic5.jpg" alt="" /></a>
        </li>
      </ul>
      <div class="arrow" id="arrow">
        <a href="javascript:;" class="prev" id="arrLeft"></a>
        <a href="javascript:;" class="next" id="arrRight"></a>
      </div>
    </div>
  </div>

  <script>
```

```
  </script>
</body>

</html>
```

正确的js代码准备

```
// 需求：点击左右按钮，切换图片的位置
// 思路：
// 1. 标签有一个数组[li, li, li, li, li]
// 2. class类名一个数组[left1, left2, middle, right2, right1]
// 3. 我们靠数组的下标，把class赋予给li，影响li的位置
// 4. css中有transition，所以class改变会有动画效果出现

// 1. 准备class的数组
var config = ["left1", "left2", "middle", "right2", "right1"];
// 2. 为了保证图片加载完成后，再进行过渡动画，才能看到图片移动的过程，所以写在onload里
window.onload = function () {
    // 3.1 获取li元素们
    var liList = document.querySelectorAll("#slide li");
    // 3.2 产生索引 - 对应class类名 - 赋予给标签
    for (var i = 0; i < liList.length; i++) {
        liList[i].className = config[i];
    }

    // 4. 实现右侧点击按钮功能
    var arrRight = document.getElementById("arrRight");
    arrRight.onclick = function (ev) {
        // 4.1 把右侧类名删除，放到最左侧(数组中)
        config.unshift(config.pop());
        // 4.2 在把数组中最新的类名 - 分别赋予给 索引对应的标签 展示
        for (var i = 0; i < liList.length; i++) {
            liList[i].className = config[i];
        }
    }

    // 5. 实现左侧点击按钮功能
    var arrLeft = document.getElementById("arrLeft");
    arrLeft.onclick = function(ev){
        // 5.1 把头部元素删除后，返回并添加到数组的末尾
        config.push(config.shift());
        // 5.2 把新的class类名数组里的 - 下标对应 - 赋予给li
        for (var i = 0; i < liList.length; i++) {
            liList[i].className = config[i];
        }
    }
}
```